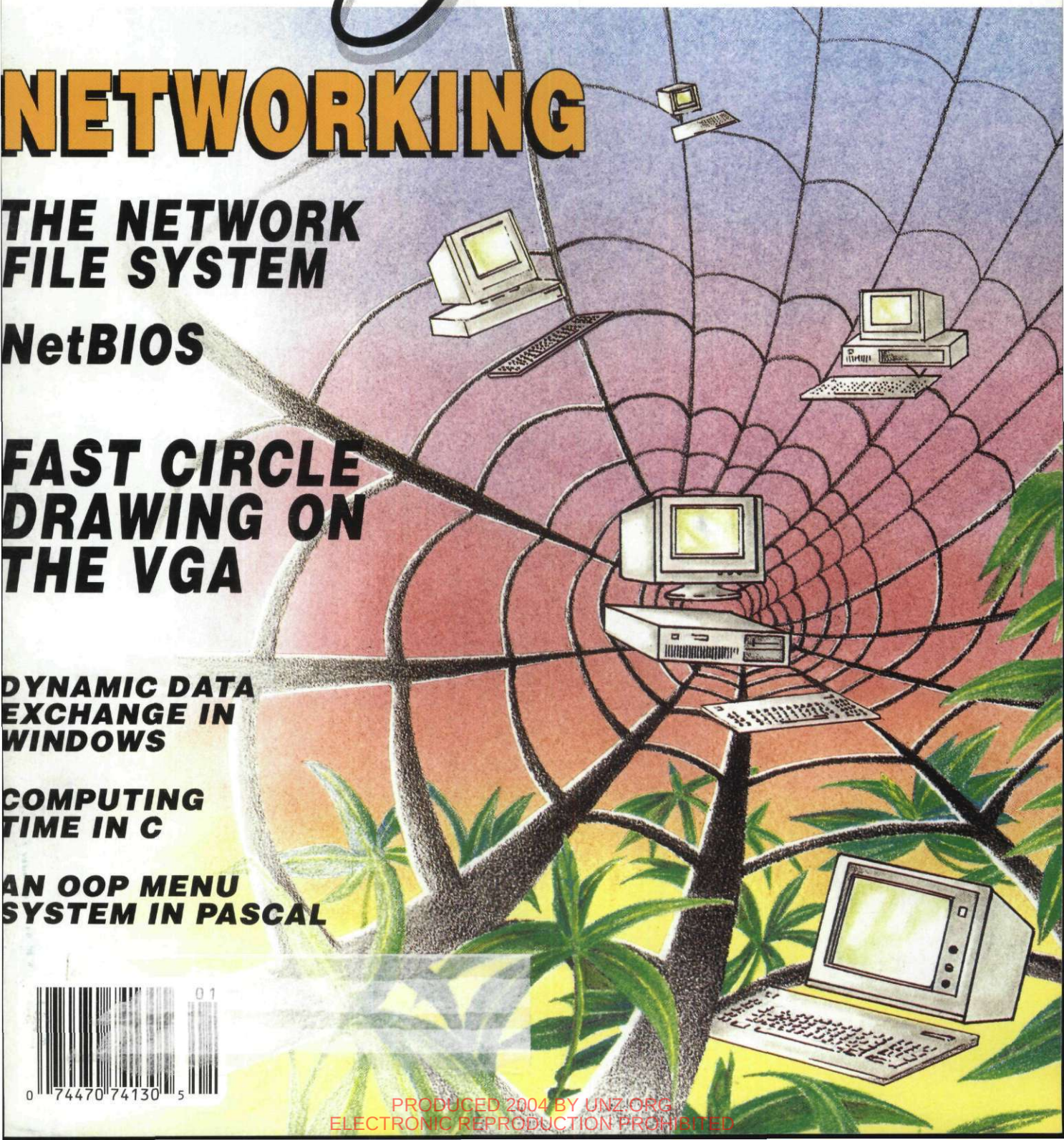
ascal • ASM • C • Graphics • Networking • OOP • Consulting

PROGRAMMER'S Journal™

The Resource Journal for IBM Micro Programmers

Qview API EATURE

January/February 1990
Volume 8.1
$4.95 Canada $5.95



NETWORKING

THE NETWORK FILE SYSTEM

NetBIOS

FAST CIRCLE DRAWING ON THE VGA

DYNAMIC DATA EXCHANGE IN WINDOWS

COMPUTING TIME IN C

AN OOP MENU SYSTEM IN PASCAL

0 74470 74130 5 01

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Now it's easy to make your programs millions of times smarter.

More and more, programmers and work station builders are using DESQview as a development tool. The reason is simple. They can create powerful, multitasking solutions today for the millions of DOS PCs in use today. Solutions comparable to those promised for tomorrow by OS/2.

The API Advantage

Programmers who take advantage of DESQview's API (Application Program Interface) get access to the powerful capabilities built into DESQview —multitasking, windowing, intertask communications, mailboxes, shared programs, memory management, mousing, data transfer, menu-building and context sensitive help.

Bells and Whistles

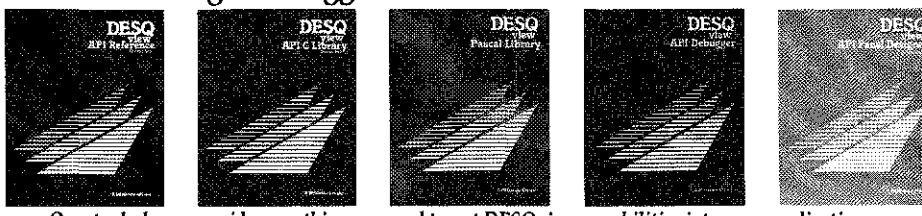
A program taking advantage of the DESQview API can spawn subtasks for performing background operations or new processes for loading and running other programs concurrently. It can schedule processing after an interval or at a certain time. It can use DESQview's intertask communications to rapidly exchange data between programs, share common code and data; or interrupt at critical events. It can use DESQview's menuing and mousing capabilities to create menus. And there's lots more it can do.

Some *of the applications under development right now using DESQview API Tools: CAD, medical systems, insurance, 3270 mainframe communications, network management, real estate, typesetting, point of sale, education, commodity trading, stock trading and online voting.*

80386 Power

Programmers can take advantage of the 80386's protected mode for large programs, yet run on DOS and multitask in DESQview—side by side with other 80386 and DOS programs. The breakthroughs that make this possible: DOS Extenders from PharLap Software and AI Architects and DESQview support of these DOS extenders.

A Complete Interactive Toolkit —Including a Debugger



Quarterdeck can provide everything you need to put DESQview capabilities into your applications.

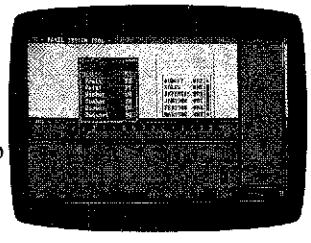
API Reference Manual The key to the power of the DESQview API, this manual contains all you need to know to write Assembly Language programs that take full advantage of DESQview's capabilities. And there's an 'include' file with symbols and macros to aid you in development.

API C Library Here are C language interfaces for the entire set of API functions. It supports the Lattice™ C, Metaware™ C, Microsoft® C, and Turbo C compilers for all memory models. Included with the C Library package is the API Reference Manual and source code for the library.

API Pascal Library The Pascal library provides interfaces for the entire set of API functions. It supports Turbo Pascal V4.0 and V5.0 compilers. Included are the API Reference Manual, source code for the library, and example programs.

API Debugger The DESQview API Debugger is an interactive tool enabling the API programmer to trace and single step through API calls from several concurrently running DESQview-specific programs. Trace information is reported symbolically along with the program counter,registers, and stack at the time of the call. Trace conditions can be specified to report only calls of interest

API Panel Designer This interactive tool hel you design windows, menus, help screens, error messages, and forms. It includes an editor that le you construct an image of your panel using simp commands to enter, edit, copy, and move text, as well as draw lines and boxes. You can then define the characteristics of the window that will contain the panel, such as its position, size, and title. Finally, you can specify the locations and types o fields in the panel.



The Panel Designe automatically generates all the DESQview API da streams necessary display and take input from your panel. These data streams may be grouped into panel libraries and stored on disk o as part of your program.

More Tools are Coming Quarterdeck is committ to adding tools as needed by our users. To that e we have been working with Ashton Tate and Buz words International on dBASE III and dBASE IV translators. And in the works, we have BASIC an DOS Extender libraries.

If you'd like hands-on experience with the API Toolkit, write for information on our next API Conference, coming this August.

Bring New Power to DOS with the DESQview API Toolkit

Quarterdeck

Quarterdeck Office Systems, 150 Pico Blvd., Santa Monica, CA 90405 (213) 392-9851 Fax: (213) 399-380

Number **147** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

"I Get Paid to Write Programs. And Now, I'm Writing Them Twice As Fast."

If you write programs for a living, get to know he Clarion Professional Developer® 2.1.

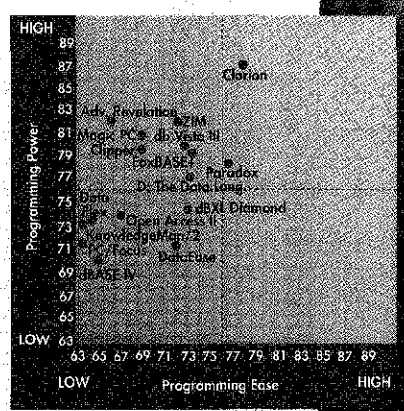
It can cut applications development time by 50% or ıore without compromising your design. It's so fast nd easy, you can complete prototypes while your lients watch.

Professional Developer 2.1 can interface to routines om C and Assembler. It allows import and export of BASE, BASIC and DIF files. Its application generator roduces commented source code which can be easily dapted with its utilities to allow you to generate .EXEs. letwork support is included at no extra cost. There are o run-time licenses or fees. And now Version 2.1 ıcludes Report Writer, which allows your clients to asily and intuitively create ad hoc reports and queries rom Clarion files, non-Clarion les or any combination).

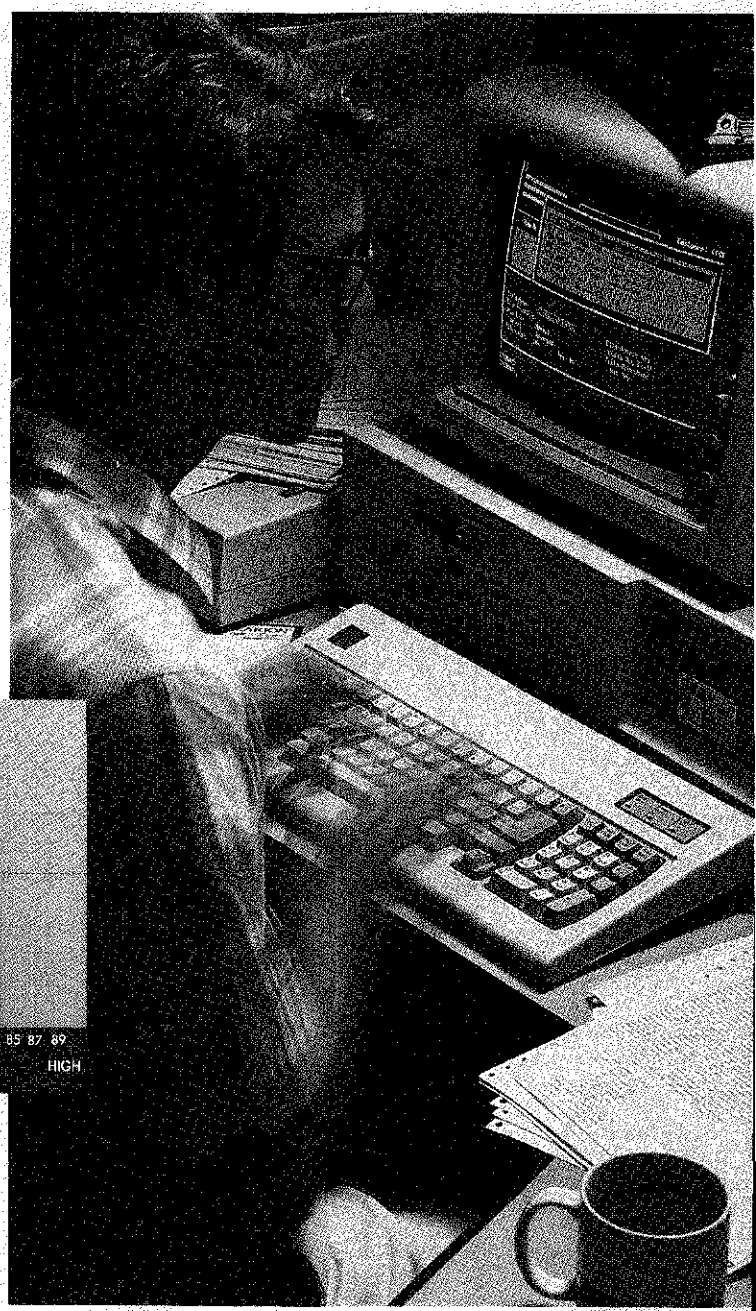
Professional Developer is an nmensely powerful, easy-to-use BMS application developer. We idn't say it: The PC WEEK ɔll of corporate satisfaction aid it. We came out head and ıoulders above the 15 other rogrammable databases. No ıe else came even close.

In this business where time money, let Professional eveloper help you generate ıore money for your time. For ɔur nearest Clarion dealer, call:

(U.S.A.) 800-354-5444
(Canada) 305-785-4555



Here's what American corporations in a PC Week survey thought of the power and ease of use of Professional Developer vs. competitors. Any questions?



PC MAGAZINE EDITORS' CHOICE NOVEMBER 15 1988 OF PROFESSIONAL DEVELOPMENT SOFTWARE

INFOWORLD Clarion Professional Developer 8.2 JULY 4 1988

CLARIONSOFTWARE™

50 East Sample Road
Pompano Beach, FL 33064

Clarion Software Products run on any IBM PC, PS/2, or true compatible with 512K of memory and a hard disk. Clarion Professional Developer, is a registered trademark and Clarion Software is a trademark of Clarion Software. Copyright 1989 Clarion Software. dBASE is a registered trademark of Ashton Tate.

Number 299 on the Reader Service Card.

SCANNED BY CANZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers

PROGRAMMER'S Journal™

Volume 8.1
January/February 1990

Contents

4 Notes from the Editor

6 Letters
Adrian B. Crum, Ken Green, Larry Smith, Steve Barber

15 Consulting: Customers You've Never Considered *by William H. Roetzheim*
Marketing your consulting services to local government agencies

20 On Graphics: Circles for the VGA *by Michael Abrash*
After meeting with Hal Hardenbergh on concepts, Michael implements fast circle drawing on the VGA

32 Networking: A Net Gain: NFS, Network File System *by Frank D. Greco*
Our UNIX correspondent takes a look at NFS for networking divergent machines

39 Networking: Exploiting NetBIOS *by W. David Schwaderer*
An overview of NetBIOS on the PC, adapted from the author's book *C Programmer's Guide to NetBIOS*

46 Windows: Dynamic Data Exchange in Windows *by William S. Hall*
Our Windows expert starts a two-part article on how to exploit the DDE protocol to move data among Windows applications

56 ASM: Using Dynamic Memory Efficiently *by John Otken*
John applies his previous columns in developing a program that illustrates effective dynamic memory use

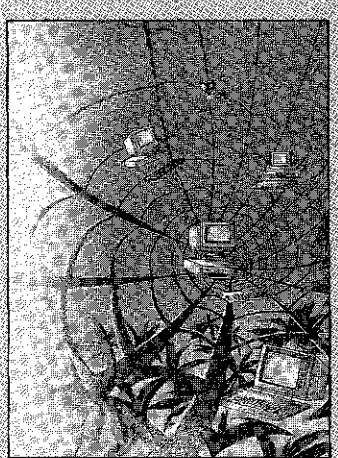
65 C: Using the DESQview API *by Todd Hoff*
Taking advantage of DESQview API features for multitasking using shared memory, semaphores, and other system services

75 C: Computing Time in C *by Murray L. Lesser*
A look at how the real world measures time, the DOS system clock and the real time clock, a bug in the DOS 3.3 CLOCK device, and a critique of the sorry ANSI C timer functions

82 Pascal: What's on the Menu? *by Tom Swan*
Tom uses the object-oriented features in Turbo Pascal 5.5 to create a self-organizing menu system, usable as a stand-alone tool and as part of his larger file viewer

94 *PJ* Billboard

96 Advertiser Index



Time was, PC users in a company could share information on their computers only by exchanging floppy disks, by printing out files, or by using each other's machines. Now with networking, programs, databases, and other files can be accessed with the press of a button rather than the trading of seats. This issue of PJ will help you connect the threads of your networking web.

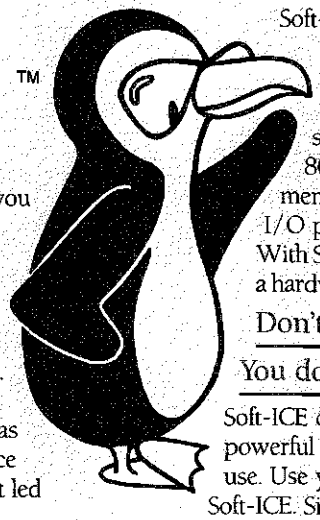
Artist:
Jarren Jester

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FINALLY. A debugging tool tough enough to handle the DOS Nasties.

New Version 2.0

Soft-ICE™



Nasty over-write? No sweat!

Soft-ICE memory range break points help you track down memory over-write problems whether you are doing the over-writing or another program is over-writing you.

Hung program? No problem!

When the system hangs, you now have hope. With Soft-ICE you can break out of hung programs no matter how bad the system has been trashed. And with Soft-ICE's back trace ranges you can re-play the instructions that led up to the crash.

Program too large? Not with Soft-ICE!

Soft-ICE runs entirely in extended memory. This means you can debug even the largest DOS programs. And since your program runs at the same address whether Soft-ICE is loaded or not you can find those subtle bugs that change when the starting address of your code changes.

System debugging? Soft-ICE is a natural!

Soft-ICE is ideal for full source level debugging of TSRs, interrupt service routines, self booting programs, DOS loadable device drivers, real-time kernels, non-DOS O/Ss and ROMs. Soft-ICE can even debug within DOS & BIOS.

How Soft-ICE Works

Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses the 80386 to provide real-time break points on memory locations, memory ranges, execution, I/O ports, hardware & software interrupts. With Soft-ICE you get all the speed and power of a hardware-assisted debugger at a software price.

Don't want to switch debuggers?

You don't have to!

Soft-ICE can run stand-alone or it can add its powerful break points to the debugger you already use. Use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated automatically.

MagicCV with Soft-ICE

Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere.

"These may be the only two products I've seen in the last two or three years that exceeded my wildest expectations for power, compatibility and ease-of-use."

—Paul Mace
Paul Mace Software

Soft-ICE	$386
MagicCV	$199
MagicCV for Windows	$199

Buy Soft-ICE & MagicCV(W) —Save $86.

Buy MagicCV and MagicCVW —Save $100.

Buy All 3 —Save $186.

30 day money-back guarantee
Visa, MasterCard and AmEx accepted

1988 BYTE Award of Distinction

New Soft-ICE 2.0 features

- Back Trace Ranges
- Symbolic & Source level debugging
- EMS 4.0 support with special EMS debugging commands
- Windowed user interface

Nu-Mega™ TECHNOLOGIES

CALL TODAY (603) 888-2386
or FAX (603) 888-2465

RUN CODEVIEW IN 8K MagicCV™

CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 to load CodeView and symbols in extended memory. This allows MagicCV to run CodeView in less than 8K of conventional memory on your 80386 PC.

NEW—Version 2.0 includes EMS 4.0 driver.
Attention Windows Developers!
Version available for CVW.

P.O. BOX 7607 ■ NASHUA, NH ■ 03060-7607

Number 160 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers
PROGRAMMER'S Journal™

Editor
M. Steven Baker
Managing Editor
Robert E. Wanetick
Contributing Editors
Michael Abrash
Frank D. Greco
Tom Swan

Contributing Writers
Michael Abrash
Frank D. Greco
William S. Hall
Todd Hoff
Murray L. Lesser
John Otken
William H. Roetzheim
W. David Schwaderer
Tom Swan

Publisher
Elizabeth Oakley
Associate Publisher
Bobbi Sinyard
Production
Ralph Bentley
Illustration
Jim Carpenter
Advertising
Robert Tappan
Jay Moore
Circulation
Eric Malin
Fulfillment
Gary Hoyer
Accounting
Lori Fanslow
Deborah Loftis
Administrative Assistant
Sylvia Long
Founder
Greg Estes

Business and Editorial Offices
Programmer's Journal
150 North 4th Street
Springfield, OR 97477
503-747-0800
800-234-0386
Fax: 503-746-0071

Subscriptions: 1 year (6 issues) $24
($30 in Canada and $40 elsewhere)
Change of Address
Programmer's Journal
P.O. Box 70387
Eugene, OR 97401

Articles and letters in *Programmer's Journal* represent the views of the authors and not necessarily those of the publisher. Letters to the editor become the property of *Programmer's Journal* and may be edited for publication.

Code and listings in *Programmer's Journal* appear "as is," and *Programmer's Journal* makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. *Programmer's Journal* assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan or Jay Moore at 800-234-0386, or write to *Programmer's Journal*, P. O. Box 70167, Eugene, OR 97401.

Programmer's Journal is produced using the following software products: Ventura Publisher,™ Aldus PC PageMaker,® and Corel Draw. The PostScript™ files are output on a Linotronic™ 200.

Programmer's Journal (ISSN 0747-5861) is published bimonthly by Oakley Publishing Company. Second-class postage paid at Springfield, OR, and additional mailing offices.

© 1990 *OAKLEY PUBLISHING COMPANY*

NOTES from the Editor

This issue of *PJ* focuses on networking, one of those truly elusive topics. For longer than I care to remember, each successive year was to be "the year of the LAN." Although these predictions were never fully realized, networked computers are becoming pervasive in the workplace. But in most situations, the network is like DOS itself, a background convenience or irritant (depending on your perspective) upon which DOS applications ride.

Two paradigms are in place for networking: dedicated file servers with isolated PC workstations, and the distributed-computing model (where literally everything can potentially be accessible). Although Novell Netware (file-server model) dominates DOS network installations, other contenders include 3COM 3+Open, Banyan Vines, SUN's TOPS and PC-NFS, Artisoft LANtastic, and now OS/2 LAN Manager. At issue with the current networking software is its use of divergent proprietary technology. The dilemma for programmers is whether to limit network programming to DOS 3.x functions or to write for proprietary network software. Although DOS 3.x network functions have considerable utility, they pale compared to that available with each custom network protocol. Still, the largest problem with current network software continues to be that the network drivers (TSRs) compete for valuable PC memory with the ever-hungry DOS application programs and user TSRs.

Another interesting development is minicomputer SQL-server database technology moving to the PC. Time will tell whether most networked PCs will mass-migrate to SQL-server technology for their database needs. This has clearly been the model in the workstation world. Note the success of Sybase, Oracle, and other SQL servers and the posturing in the DOS world with these same companies.

COMDEX a-Cometh

When this editorial goes to print, I will be at COMDEX in Las Vegas. I must admit that COMDEX always "blows me away"—a small-town boy, I still get "big eyes" in Las Vegas watching companies drop half a million in a week's time on marketing and PR. This year, the spotlight will be on the 80486 and the new EISA boxes. Also, some real OS/2 applications will have arrived—not the vaporware displayed under the fluttering OS/2 banners a year ago. The 386 version of OS/2 is getting solid fast, with Developer's Kits to be available about when you read this. OS/2 Lan Manager 2.0 is also hot off the press.

The PC world is also supposed to be aflutter with the concept of PM Lite, a DOS API that looks like Presentation Manager. Cyco (Atlanta, Georgia), the developer of highly rated AutoCAD add-ons, has also announced a similar product for DOS programmers: a DOS subset library of most important OS/2 PM functions. Both PM Lite and the Cyco Presentation Manager would allow writing PM applications that could be compiled and linked for DOS machines. Unfortunately, these products made much more sense a year ago.

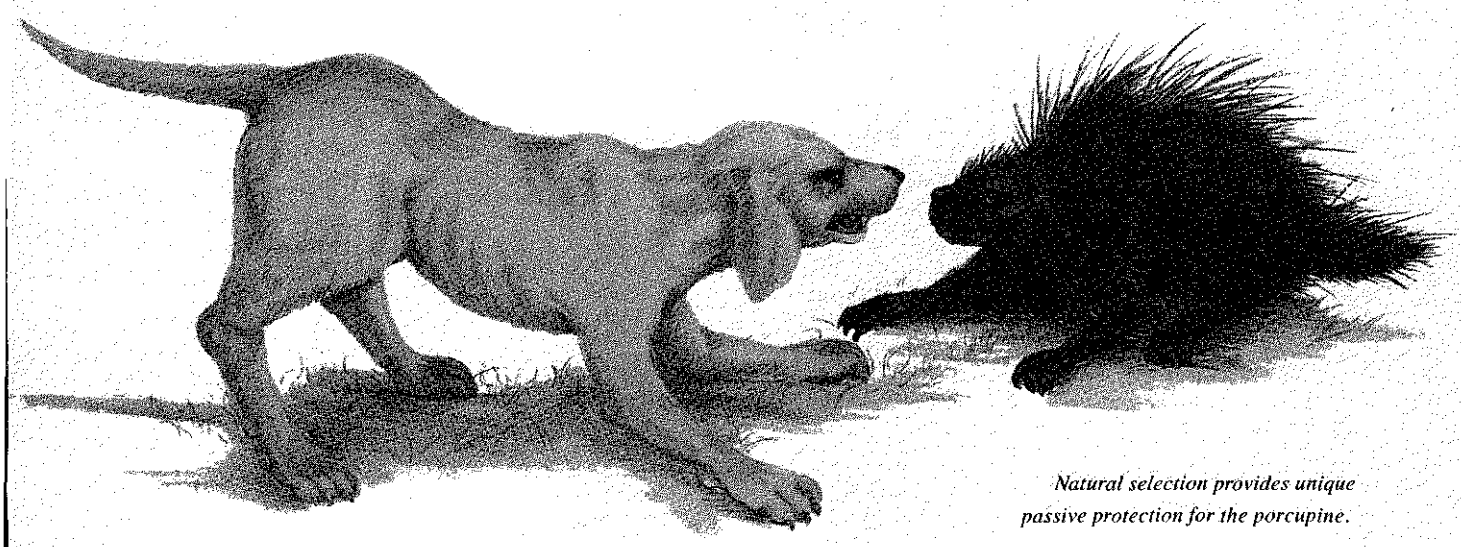
The 80486 NOW?

Of course, the big story at COMDEX will be the 80486. When I was inking my editorial for the previous issue of *PJ*, I was very close to going out on a limb with intuition and some inside information. I thought that the 486 chip would not really ship until 1990. Unfortunately, I "held my pen," which was clearly the safe thing to do since Intel folks kept up their front so well. I knew at the time that the 80486 B-step chips were getting close to "cleanliness," but a number of bugs remained. The question was whether Intel would respond to the pressure from hardware vendors and deliver questionable chips for all those 486 boxes waiting for their "brains" (CPUs) or bite the bullet and await a clean chip mask.

To remind you, several versions of both the 80286 and 80386 were shipped in volume prematurely, with potentially nasty bugs. There must still be several hundred thousand 386 machines that will hang when operated in the protected mode using the 80387 (infamous Errata 21). Fortunately, most of these machines just run DOS, and so this problem won't be discovered by users until the next software upgrade. I heave a sigh of relief that Intel decided not to replicate the 286 and 386 morass.

COMDEX should have plenty of 80486 EISA boxes. Sadly, neither EISA nor IBM's MCA bus deals with a major PC bottleneck: dumb peripherals that require CPU time. Run a high-speed modem program multitasking on your fast 486 box and watch your PC slow to a snail's pace. I guess you can't win them all. —*Steve*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Natural selection provides unique passive protection for the porcupine.

The Activator - Natural Selection For Software Protection

Inventor and entrepreneur Dick Erett explains how "The Activator" provides sane protection for your intellectual property.

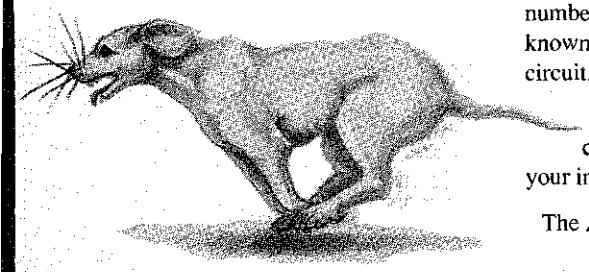
"In any industry, just as in nature, the process of natural selection raises one solution above another. Natural selection is the most elegant of engineers.

In the area of software protection The Block has been selected by the marketplace as the solution that works. Over 500,000 packages are protected by our device.

For the past 4 years our philosophy has been; *'You have the right and obligation to protect your intellectual property.'*

A New Ethic For Software Protection

In allowing end-users unlimited copies of a software package and uninhibited hard disk and LAN operation, The Block has created a new ethic for software protection.



By removing protection from the magnetic media we remove the constraints that have plagued legitimate users.

They simply attach our key to the parallel port and forget it. It is totally transparent, but the software will not run without it.

A New Technology For Software Protection

Our newest model, The Activator, builds on our current patented design, and establishes an unprecedented class of software protection.

We have migrated and enhanced the circuitry of The Block to an ASIC (Application-Specific Integrated Circuit) imbedded in The Activator.

This greatly improves speed and performance, while reducing overall size. Data protection can also be provided.

Programmable Option

The Activator allows the software **developer** the option to program serial numbers, versions, or other pertinent data known only to the developer, into the circuit, and access it from the program.

Once you program your part of the chip, even we have no way to access your information.

The ASIC makes emulation of the device virtually impossible. It also presents an astronomical number of access combinations.

Full 100% Disclosure

Since The Activator is protected by our patent we fully disclose how it works. Once you understand it, endless methods of protection become evident.

Just as no two snowflakes are the same, no two implementations of The Activator are identical. And like the snowflake the simplicity of The Activator is its greatest beauty.

We never cramp your programming style or ingenuity. Make it as simple or complicated as you desire.

Let us help safeguard what's rightfully yours. Please call today for additional information or a demo unit. *It's only natural to protect your software."*

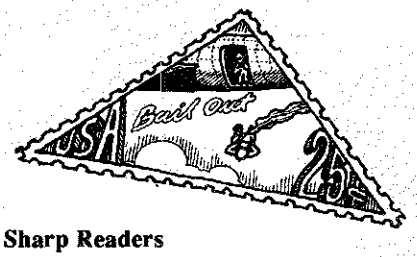
1-800-333-0407 ext.103
In Connecticut 203-329-8870
Fax 203-329-7428

Software Security inc.

1011 High Ridge Road
Stamford, CT 06905

Unlimited Copies • Programmable • No Batteries • Small Size • Fast • Patented • Data Protection

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Sharp Readers

Mark E. Huss's observations in the "Letters" section of *PJ* 7.6 are correct. The solution is to use the smallest shift value if a character appears more than once in the pattern. Although Mark's code fixes the problem, there is one more discrepancy, which another reader, Jon Grewer from Tustin, California, called me to point out:

The SCASB instruction, which is used to get the shift value, performs a character-by-character comparison. Therefore, my statement that only 9 comparisons

```
; SEARCH.ASM    A STRING SEARCH ROUTINE
; (C) 1989 BY ADRIAN CRUM
;
; IMPLEMENTS A BOYER-MOORE ALGORITHM
;
;@CALLING PARAMETERS:
;
;     BX = OFFSET OF PATTERN TO SEARCH FOR
;     CX = LENGTH OF PATTERN TO SEARCH FOR
;     DX = OFFSET OF LAST BYTE OF TEXT TO BE SEARCHED
;     SI = OFFSET OF FIRST BYTE OF TEXT TO BE SEARCHED
; ES:DI = OFFSET OF 512 BYTE BUFFER TO BE USED FOR SHIFT TABLE
;
; RETURNS CARRY CLEAR AND SI=MATCHING TEXT OR CARRY SET IF NO MATCH.

SEARCH  PROC

        PUSH    BX                      ; Save these for exit
        PUSH    CX
        CMP     CX,2
        JB      NOT_FOUND               ; Skip single char patterns

INIT:   PUSH    DI
        MOV     AX,CX
        DEC     CX                      ; CX = pattern length - 1
        PUSH    CX                      ; Save CX
        MOV     CX,256                  ; Shift table length / 2
        CLD
        REP     STOSW                   ; All table entries = pattern length
        POP     CX                      ; Restore CX
        POP     DI                      ; Restore table pointer
        PUSH    CX                      ; Save CX again
        PUSH    SI                      ; Save text pointer
        PUSH    BX                      ; Save pattern pointer
        MOV     SI,BX                   ; SI points to pattern

BUILD:  MOV     BL,[SI]                 ; Get a char from pattern
        XOR     BH,BH                   ; Clear upper half of BX
        SHL     BX,1                    ; Convert to offset in shift table
        MOV     ES:[BX+DI],CX           ; Put shift value in table
        INC     SI                      ; Increment pattern pointer
        LOOP    BUILD                   ; Process next pattern char
        MOV     AL,[SI]                 ; AL contains last char in pattern
        POP     CX                      ; CX points to pattern
        POP     SI                      ; Restore text pointer
        POP     BX                      ; BX = pattern length - 1

COMPARE:
        CMP     SI,DX                   ; Reached end of text yet?
        JA      NOT_FOUND               ; Yes, bail out
        CMP     AL,[BX+SI]              ; Last char in pattern match?
        JNE     SHIFT                   ; No, shift pattern
        PUSH    SI                      ; Save the pointers
        PUSH    DI
        PUSH    CX
        MOV     DI,CX                   ; DI points to pattern
        MOV     CX,BX                   ; Put pattern length-1 in CX
        REPZ    CMPSB                   ; Compare text to pattern
        POP     CX                      ; Restore pointers
        POP     DI
        POP     SI
        JE      FOUND                   ; Match found, bail out

SHIFT:  PUSH    BX                      ; Save the pattern length
        MOV     BL,[BX+SI]              ; Get mis-matched text char
        XOR     BH,BH                   ; Clear upper half of BX
        SHL     BX,1                    ; Convert to offset in shift table
        ADD     SI,ES:[BX+DI]           ; Add shift value to text pointer
        POP     BX                      ; Restore pattern length
        JC      NOT_FOUND               ; If segment wrap, bail out
        JMP     COMPARE                 ; Otherwise, loop

FOUND:  POP     CX
        POP     BX
        CLC                             ; Search successful
        RET

NOT_FOUND:
        POP     CX
        POP     BX
        STC                             ; Search unsuccessful
        RET

SEARCH  ENDP
```

Shove These In Your PC!

Pop POWERLINE II cards into your PC and drive them with our COMMANDO software. Get the results you want from your VOICE PROCESSING APPLICATIONS.

- Voice Messaging
- Call Processing
- Audiotex
- Telemarketing
- Order Processing
- Call Distribution

Our POWERLINE II multiple-port board functions as the heart of a Voice Communications System based on the IBM PC/XT/AT/386 architecture. Each card provides dedicated hardware support for voice processing, call analysis, touch-tone & rotary interaction, and telephone/PBX/Centrex integration.

COMMANDO's extensive function library allows you to rapidly generate sophisticated inbound and/or outbound applications supporting as many as 16 lines—even in the background!

And, all Talking Technology products carry a 2 YEAR WARRANTY. We design them to last!

POWERLINE II $599.00
COMMANDO $139.00
SINGLE LINE from . . . $295.00
VISA/MC/AMEX/COD

FOR ADDITIONAL INFORMATION OR SALES CALL:
(415) 652-9600
FAX (415) 652-5311

TALKING TECHNOLOGY, INC.
4383 PIEDMONT AVE., OAKLAND, CA 94611

Number 297 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

were made is untrue. (Actually, my statement was correct for the original algorithm—it's my implementation that makes it untrue.) Accounting for the SCASB comparisons yields 19 comparisons—aaargh! The corrected routine is provided *[in the box on page 6]*.

Adrian B. Crum
Fontana, California

Keeping Up to Date

John Otken's article ("A Library System for Assembly", *PJ* Vol. 7.5) was very interesting to me as I have about 150 routines and had never figured a way to manage them properly—I certainly didn't want a directory of 150 ASM files and 150 OBJ files or to have to recreate the library every time one little thing changed.

However, some of those ASM files are dependent on other files; so, I did need some sort of MAKE capability so that if a control file was changed, the dependent file would be reassembled.

Unashamedly stealing some of John's ideas, I created a system that would keep my library up to date, allow for those file dependencies, avoid multiple calls to the assembler and librarian, and eliminate file "litter." This system uses the Turbo Assembler (TASM) and its utilities, but it can readily be adapted to MASM or OPTASM.

This system is controlled by REFRESH.BAT *[Listing 1 in the box on page 10]*, and it does the following:

1. Makes a list of ASM files that need assembling using ASMLIST.EXE.
2. Assembles those files using TASM.
3. Assembles any files having dependencies (using MAKE).
4. Makes a list of files that need to be updated in the library.
5. Changes the library using TLIB.
6. Cleans up the extraneous files (i.e., deletes all OBJ and WKG files).

The list of files needing assembly, named NEEDASM.WKG, is generated by the program ASMLIST.EXE *[Listing 2 on the code disk for this issue]* and is simply a list of ASM files newer than the main library file.

The list of library update files (ADDTOLIB.WKG) is created by LIBLIST.EXE *[Listing 3 on the code disk]* and is all of the OBJ files on the directory. (As OBJ files are not retained, they will exist only because they've just been assembled.) This is needed because the librarian won't take wildcard arguments—"*" means something else.

Continued on page 10

STAGE#1 -Action Diagram Editor

STAGE#1 is a full screen editor that automates the generation and editing of action diagrams, the most versatile diagramming method available today. Action diagrams can be used to define software structure from the top-most overview down to the lowest level pseudo code. No other diagramming method has that capability.

More than 70 commands provide the familiar word processor functions, plus a few unique features. For example:

CONVERTS ACTION DIAGRAMS;

```
*Print_Character_Set

   For line Ø to 7
      Put leading spaces in the output string
      Put the line number and ': ' in the output string
      Calculate offset (line * 32)

      If line is not Ø or 4

         For count equals Ø to 31
            Character value is count plus offset
            Add character and space to the output string
      /* Endfor */

      /* Else */
         /* Add 'Control characters, can't print them!' to the */
           /* output string */
      /* Endifelse */

      /* Display the output string */
      /* Add CR and LF to the output string */
      /* Print the output string */
   /* Endfor */

/* Print a formfeed */
```

INTO COMMENT LINES!

STAGE#1 also doubles as a normal text editor for entering program code. Direct access to DOS allows trial compilation or assembly without exiting the editor.

System requirements: IBM PC/XT/AT, PS/2 or compatible; PC or MS-DOS, 2.1 or higher; minimum of 256KB of memory; CGA, EGA, VGA or monochrome monitor; two disk drives; printer with either the full ASCII or the IBM graphics (line-draw) character set. A plotter is not required.

INTRODUCTORY PRICE $159.00

Price: ~~$189.00~~ (US), includes shipping (UPS ground) within the continental US. Washington residents add 7.8% sales tax. Educational discounts and site licensing are available. NOT copy protected. Specify 3 1/2 or 5 1/4 inch disk.

Why settle for a program that will produce only flowcharts and organization charts when you can have much more?

St Helens Software

SIMPLY POWERFUL

VISA MasterCard

Mt. St. Helens Software
P.O. Box 3319
Pasco, Washington 99302-3319

For orders or information:
(509) 547-2582
8 AM - 5 PM (PST)

St. Helens Software, Simply Powerful and STAGE#1 are trademarks of Mt. St. Helens Software. Other brand and product names are trademarks of their respective holders.

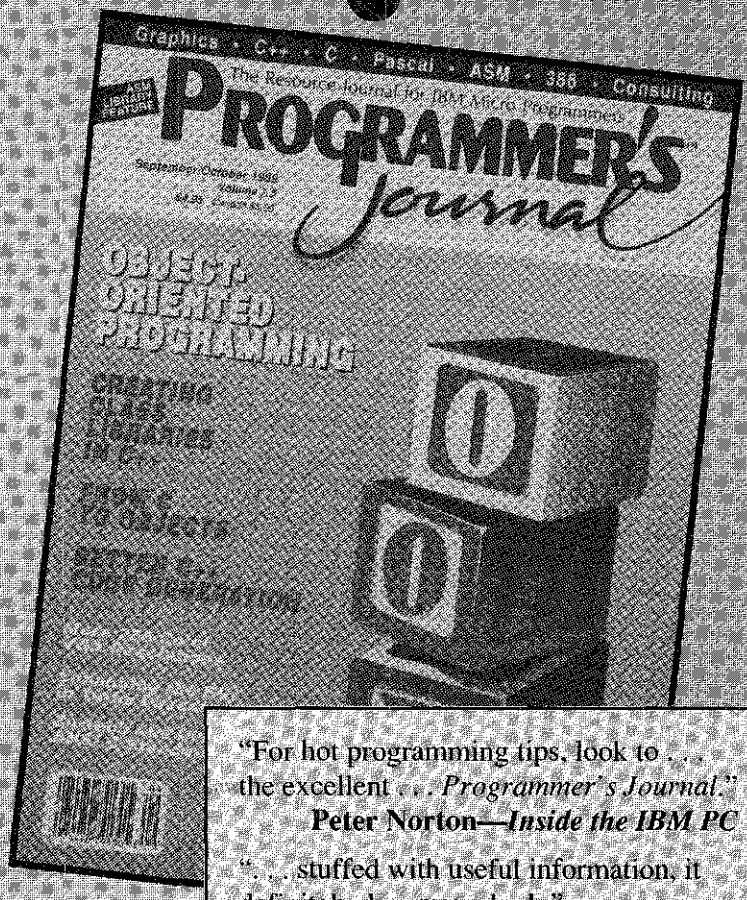
©Copyright 1989 Mt. St. Helens Software

Number 319 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

"...love this magazine!"

Industry experts agree that when it comes to a truly useful programming magazine, ***Programmer's Journal*** just can't be beat. Recognized by the **Computer Press Association** in 1989 for excellence in journalism, ***PJ*** is ". . . full of practical information and guidance presented with superb clarity and perspective." Try ***PJ*** and judge for yourself. It's easy—**just call toll free** to start your subscription.



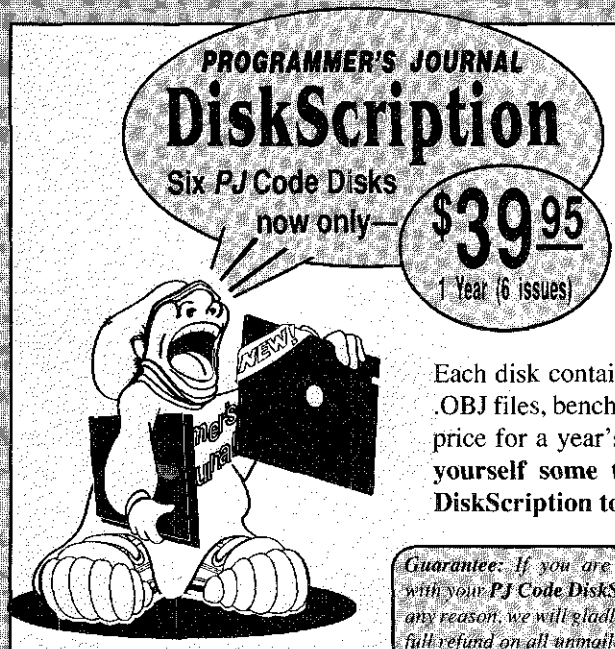
"For hot programming tips, look to . . . the excellent . . . *Programmer's Journal*."
Peter Norton—*Inside the IBM PC*

". . . stuffed with useful information, it definitely deserves a look."
Ray Duncan—*Dr. Dobb's Journal*

"I really love this magazine!"
Paul Mace—Paul Mace Software

To subscribe call toll free:

800-234-0386



All Our Code— On Disk

Typing in long code listings uses up a lot of valuable time. And calling up bulletin boards is not exactly a free ride. So what does the modern programmer do? Get a ***PJ* Code DiskScription,** of course!

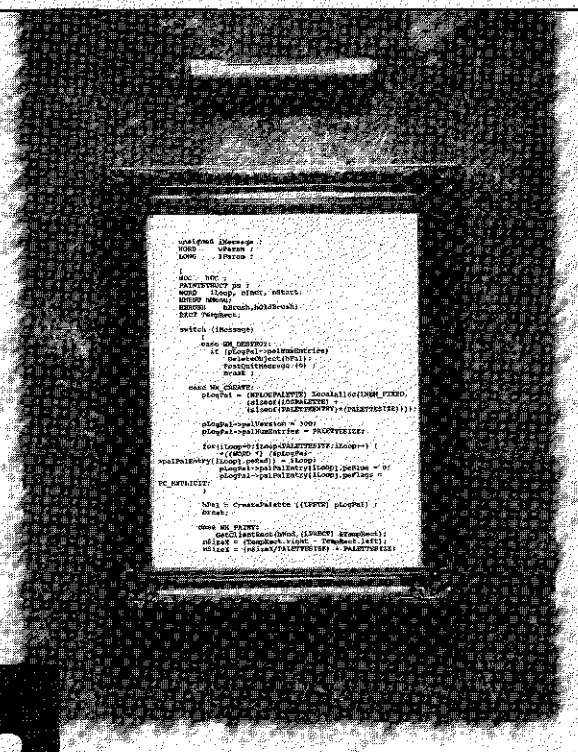
Each disk contains all of the program listings published in ***PJ*** plus .EXE files, .OBJ files, benchmark source files and more! If purchased separately the regular price for a year's worth of ***PJ*** code (6 disks) would be $60.00. **You can save yourself some time, effort and over $20.00 by ordering your *PJ* Code DiskScription today!**

*Guarantee: If you are dissatisfied with your **PJ Code DiskScription** for any reason, we will gladly send you a full refund on all unmailed diskettes.*

All *PJ* Code DiskScriptions must be prepa
Return the card on the opposite page with payment. Or, for faster service with your VISA MC call toll-free: 800-234-0386

P. O. Box 70387 • Eugene, Oregon 97401 • (503) 747-0800

PRODUCED BY JOLT UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



We appreciate fine art.

If you really want your software appreciated, join Microsoft.

Our mission is to make our system and application software tighter, faster and more elegant. So fine-tuned, optimized code is admired more now than ever.

And, you'll be working with the masters. We're home to some of the greatest software artists of all time.

Software Design Engineers

We're offering you a chance to do things you can't do anywhere else. Become involved with everything from object-oriented methodology, operating systems, and networking to user interfaces, performance evaluation, sophisticated workgroup applications, graphical word processors and more. And we're planning on continuing to break new ground in the future.

To pull this off you'll need: Programming expertise and a background that includes micro's, C or Pascal, 8086, 68000, UNIX™/XENIX, Macintosh Toolbox, and/or MS-DOS.

Program Managers

Actually, "Technical Catalyst" would be more accurate. Because we believe engineers are the best people to envision and design product features. To create the user and programmer interfaces. To write the specs, and drive them through to release.

If you'd like to impact the agenda for the future Microsoft products, bring your impressive technical skills and unique ability to make things happen to Microsoft.

We also expect you to bring an in-depth knowledge of the industry and a background in graphical environments, operating systems, languages, multimedia or applications.

What would you appreciate?

We don't believe an artist has to suffer. You'll be working with some of the true masters in software; using the absolute latest equipment; and doing it all in the scenic beauty of the Pacific Northwest.

To apply, please send your resume in confidence to: MICROSOFT CORPORATION, Recruiting, Dept. CWART-0190PJ, One Microsoft Way, Redmond, WA 98052-6399. No phone calls, please. We are an equal opportunity employer.

UNIX is a registered trademark of AT&T.

Microsoft

HARDCORE SOFTWARE

Number 232 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

IF YOU PROGRAM WITH A MICROSOFT LANGUAGE THEN SHAKE HANDS WITH....

WINDOWING POWER!

MXW

WINDOWING LIBRARY

Fast, Powerful, Versatile Windowing Actions You Can Call From BASIC, C, Pascal, FORTRAN, QuickBASIC, QuickC, and QuickAssembler

- Exclusive "TAILORING" concept provides FORMATTING AND COLOR CONTROL OF EACH TEXT LINE AND ELEMENT. Create MULTIPLE WINDOW Setups.
- Select from a VARIETY OF WINDOW DESIGNS - Standard, PopUp, ZoomUp, SlidePanel, FanOut, and others.
- SAVE/RESTORE UNLIMITED FULL or selctd screen areas - RESTORE SELCTD SCREEN AREAS ANYWHERE!
- USER PROMPT DEVICES - MENUBAR, MENUARROW, FIRST LETTER HIGHLIGHT.
- In System handling of USER MENU SELECTIONS - Cursor Line, Key Press info. returned.
- Create FULL BLOWN TOP SCREEN MENU SYSTEMS with P/Downs, etc.
- BAR MENUS on ANY row from 1-25.
- JUSTIFY Window Text (Option).
- SPECIAL EFFECT Screen/Window Clearing; explode, UpDown, etc.
- Variety of BORDER, HEADER, and 3-D SHADOW EFFECTS.
- Create and INTERMIX invoice, entry, and other forms with menus, messages, etc.
- READ Video, Hardware, Disk, Drive, Equipment Stats with powerful BIOS/DOS Interface.
- Comprehensive CALENDARING SYS. Procedures to inc. in programs.

MXW Windowing Sys. $59.95

SYSTEM HIGHLIGHT DEMO AVAILABLE - $2.95 (deducted if MXW purchased)

Comprehensive Manual/Reference & Source Code Sample Modules on disk to reduce cost.

(516) 434-1979

Send Check or Money Order to:

MTD SOFTWARE DEVELOPMENT, LTD
21 RALPH AVENUE
NORTH BRENTWOOD, NY 11717

Microsoft, "Quick" as used, are Registered Trademarks of the Microsoft Corporation

Number 312 on the Reader Service Card

Both lists, however, also include any command arguments needed to be given to the assembler or librarian.

The MAKE file is a bit tricky. Assume that a file, DEFS.INC, contains some definitions used in MYSTUFF.ASM. If we make a change to DEFS.INC, we want MYSTUFF.ASM to be reassembled. We would create a file (Turbo Make's default file name is MAKEFILE) as follows:

```
MYSTUFF.OBJ: MYSTUFF.ASM DEFS.INC
    TASM /z MYSTUFF.ASM
```

The MAKE program looks in MAKEFILE and sees that MYSTUFF.OBJ is dependent on any changes made to MYSTUFF.ASM or DEFS.INC. If either file's date is later than MYSTUFF.OBJ, MYSTUFF is reassembled creating a new MYSTUFF.OBJ with, of course, a later time and date.

That, however, defeats the purpose of not keeping a lot of extraneous OBJ files around. So, in this system I cheat a little, and my MAKEFILE would contain the following:

```
MYSTUFF.ASM: DEFS.INC
    TASM /z MYSTUFF.ASM
    TOUCH MYSTUFF.ASM
```

The MAKE program assembles MYSTUFF.ASM only if DEFS.INC is a later file. Then, TOUCH updates MYSTUFF.ASM's date and time. One warning: The TASM command line cannot contain errors, or TASM will erase MYSTUFF.ASM. (Of course, that's easily recovered from the backups—right? Right?)

Note that if MAKEFILE doesn't exist, REFRESH.BAT skips the MAKE step.

To test this system, create REFRESH.BAT and the two C programs. Add a couple of ASM utilities (or John's two ASM utilities, MS_DOS.ASM & OUTCHR.ASM), preferably on a blank directory. Make sure that the TASM, TLIB, MAKE, and TOUCH programs (or your equivalents) are in the path.

Type "REFRESH"; you'll see the ASM programs assembled and added to the new library, ROUTINES.LIB. You won't see the two WKG files unless you comment out the DEL *.WKG line in REFRESH.BAT, but ROUTINES.LST will give you a brief list of the library's contents.

Note that the two C programs use defines; they can easily be changed to include your own filenames and arguments.

My thanks to John for suggesting a cleaner way to control things and my thanks to PJ for its help over the years. I hope this is useful to others.

Ken Green
Diamond Bar, California

```
LISTING 1

ECHO OFF
REM     *****   Updates the library file of ASM routines     *****

REM     1. Make a list of ASM files that need assembling using ASMLIST.EXE.
ASMLIST
REM         End if there was a problem
IF ERRORLEVEL 1 GOTO End
REM         If no NEEDASM.WKG file exists, nothing needs changing
IF EXIST NEEDASM.WKG GOTO Step2
ECHO No changes needed.
GOTO End

REM     2. Assemble those files using TASM.
:Step2
TASM /z @NEEDASM.WKG
IF ERRORLEVEL 1 GOTO End

REM     3. Assemble any files having dependencies (using MAKE).
IF NOT EXIST MAKEFILE GOTO Step4
MAKE

:Step4
REM     4. Make a list of files that need to be updated in the library.
LIBLIST
IF ERRORLEVEL 1 GOTO End

REM     5. Change the library using TLIB.
TLIB ROUTINES.LIB @ADDTOLIB.WKG, ROUTINES.LST

REM     6. Clean up the extraneous files (i.e. delete all .OBJ and WKG files).
DEL *.WKG
DEL *.OBJ
:End


[NOTE: Because of space limitations, Listings 2 and 3 are not printed in
this issue. They are available on the Code Disk for this issue as well as
through our bulletin board services (see pages 16 and 88). -Ed.]
```

Continued on page 93

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE ULTIMATE PROTECTION

StopCopy Plus and StopView:
Quite Simply, the Best Ways to Protect Your Valuable Software Investment.

StopCopy Plus:
Sophisticated Copy Protection that defeats all hardware/software copiers, is loaded with developer options, and remains completely transparent to the honest end user.

StopView:
Advanced copy deterrent that stops hackers, allows tracing of your software, and provides many options without copy protection.

Free demonstration disk available!
Call Today!

BBI Computer Systems
14105 Heritage Lane
Silver Spring, MD 20906
Phone: (301) 871-1094 Fax: (301) 460-7545

Number **303** on the Reader Service Card

C++ Tools

Use one and two dimensional arrays as easily as built-in types!

Full source code is included, so you can see how C++ works, while you improve your productivity!

• The ***Rogue Wave C++ Toolbox*** is a set of vector and matrix classes that can dramatically simplify your programming! Working with vectors and matrices is notoriously difficult in C. With the *Rogue Wave Toolbox*, you can add two vectors as easily as two doubles! • All of the standard C arithmetic and assignment operators, like "+" and "+=" as well as all the standard C math functions, have been extended to include vectors, matrices, and a complete ***complex number class***. • We have provided *many* ***new*** functions that have been specifically designed for statistics, data analysis, and numerical modeling applications. • Vectors and matrices of integers, single and double precision floating point, and complex numbers are included. • ***Fast Fourier Transform*** server classes allow you to take the FFT or inverse FFT of any length series (real or complex) quickly and easily. Vector ***slices*** supported.

• **Improve your productivity** by concentrating on the problem you are trying to solve, rather than on programming details. • C++ is a strongly typed language. Our extensions to it allow you to do complicated operations confidently in a few lines of code. Most errors are caught at compile time! • You don't give up program efficiency either, because low-level operations (like multiplying two vectors together) have been optimized in a transparent way. • Using the inheritance property of C++, you can create new classes from our classes to do your specialized task. We show you how!

$150
(plus $3 shipping;
Visa/MC accepted)

Full Source Code; 140 page manual;
Numerous examples and test programs;
No Royalties; Compiles with Zortech
1.07 and 2.0 and many Unix compilers;

Rogue Wave; P.O. Box 85341; Seattle, WA 98145; 206-523-5831

Number **324** on the Reader Service Card

NEW BBS

SCREEN MANAGER
P·R·O·F·E·S·S·I·O·N·A·L

ADVANCED WINDOWING

• The Screen Manager Professional gives your applications the finest bullet proof windowing interface available. Although other products may boast of hundreds of functions, they often involve needless repetition. Our functions are elegant and powerful. Each accomplishes a variety of tasks with one command. All window functions are written in hand-optimized assembly language to provide the fastest possible response and smallest possible size.

ORDER

•30 day Money Back Guarantee
•Screen Manager Professional $250
•Source Code $100
•Specify Compiler when ordering •Include UPS shipping: ground $5, overnight $20, 2nd day air $10, outside U.S. $30, COD $5. •Visa, Mastercard, check or money order. •Payment must be in U.S. dollars drawn on a U.S. bank. •PA residents add 6% sales tax.

POWERFUL

• Amazing Speed
• Incredible Menus
• Context Sensitive Help Editor
• OS and Compiler Independent
• Window Functions are 100% Assembly
• Keyboard Filtering for Data Entry Systems
• Full Video Adapter Support
• Compact Linkable Library
• Background Processing
• Reconfigurable Memory Allocation Function

VERSIONS AVAILABLE FOR

•Microsoft C •Zortech C++
•Quick C •Turbo C •Lattice C •
Watcom C

AMAZING

"...an amazingly complete collection of window commands." "The support functions are easy to use and very powerful. This library warrants serious consideration."
Computer Language-May '89

• Free yourself to concentrate on the substance of your program. Over 130 functions provide all the windowing tools needed to develop serious applications whether you are a professional or a hobbyist.
• The Screen Manager Professional can help you create easy to use, visually stimulating, programs with minimum effort. Now powerful windowing is available to you at an affordable price.

NEW BBS

Demos, Software, Tech Support- A forum for the Professional Programmer
BBS (814) 234-1881

Logical Alternatives Incorporated
Calder Square, P.O. Box 10674
State College, PA 16805-0674
(814) 234-8088 FAX (814) 234-6864

Number **246** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Our Overlay Tools Have Helped A Lot of Big Companies Get Smaller.

And now we'd like to help yours. Because you know your product is only as good as its overlay structure. And when your overlay structure is too big or too slow, your product takes the heat. Enter Overlay Architect and Overlay Optimizer, the overlay problem solvers.

Overlay Architect analyzes your object code and builds a very compact Plink86Plus or .RTLink overlay structure. It can handle up to 16,000 modules on a standard 640K PC and runs in a fraction of the time it takes to link.

Overlay Architect works with just about any compiler. It produces nested overlays and multiple overlay areas, and offers support for indirect calls, multiple entry points and libraries. Additionally, it can detect unused functions and modules.

Overlay Optimizer is the performance tool you've been waiting for; it can tune any overlay structure (whether you've used Overlay Architect or not), making your overlaid program run as fast as possible in a size you specify.

Want to talk overlays with people who know overlays?

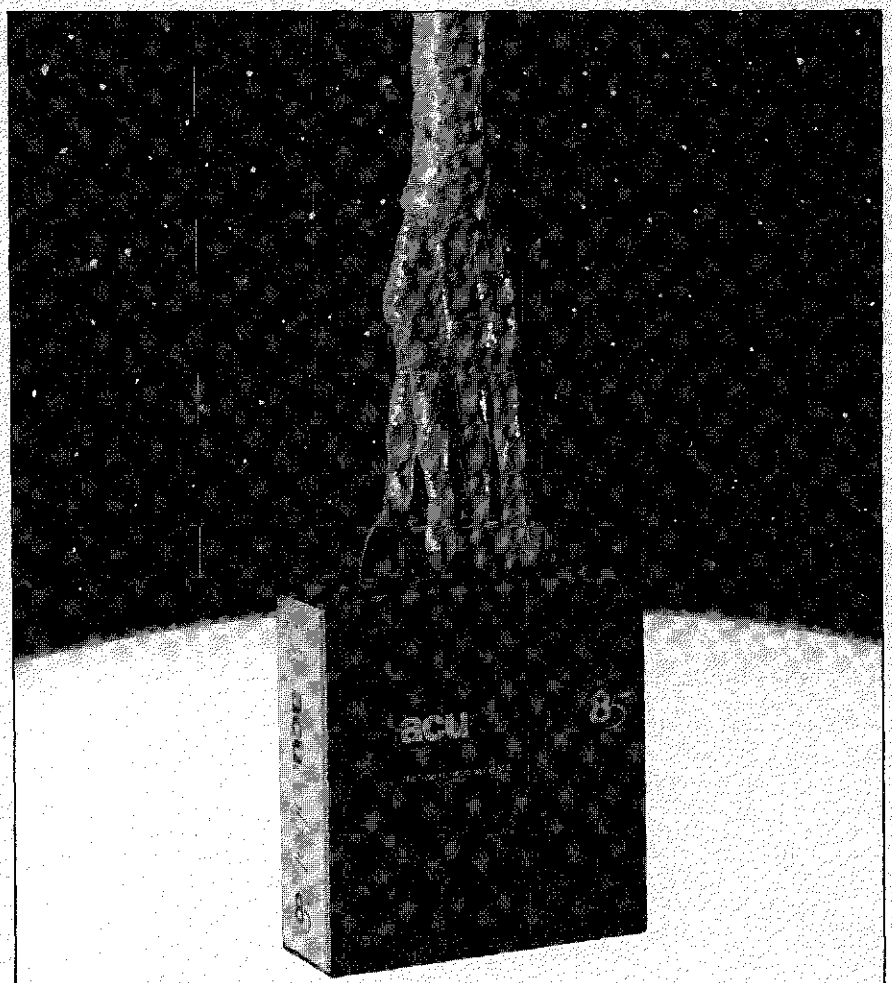
Call 303-938-1210

Overlay Architect: $369
Overlay Optimizer: $269
Purchase them together for $569
30 day money-back guarantee
We accept MasterCard/Visa

AT LAST!
SOFTWARE INC.
449 Mountain View Road
Boulder, CO 80302

We Build Software Tools That Work!

Number 332 on the Reader Service Card



INTRODUCING THE MOST PORTABLE COBOL IN THE UNIVERSE

Go where you want to go
Transport COBOL programs to almost any platform—without changing one line of code. Run on AIX, MS-DOS, NETBIOS, UNIX, OS/2, ULTRIX, XENIX and VMS without recompiling. No other COBOL-85 offers such hardware independence, or the instantly-expanded possibilities.

Open new windows in your applications
Create better-looking programs with windows. Add help screens, pop-up/pull-down menus, context-sensitive messages, more. ACUCOBOL-85 has a Display Window verb, including extensions for boxed, reverse-video and pop-up windows in color. Paint special forms on the screen with our Display Line verb. And scroll up/down inside windows.

Don't get lost in space:
ACUCOBOL-85 keeps you on course
ACUCOBOL-85 is NBS certifiable error-free under the High 1974 and Intermediate 1985 ANSI standards. If you lose course, it has its own interactive source code debugger that's fast and easy to use.

Step into the future
Make your job simpler and stay competitive—even ahead of the game. Put the next generation COBOL to work for you today.

Free evaluation copy
Call (619) 271-7097 today for your free evaluation copy of ACUCOBOL-85. And see the shape of things to come.

acucobol inc

Voice: (619) 271-7097 **Fax: (619) 566-3071**
7950 Silverton Avenue **Suite 201** **San Diego** **California** **92126**

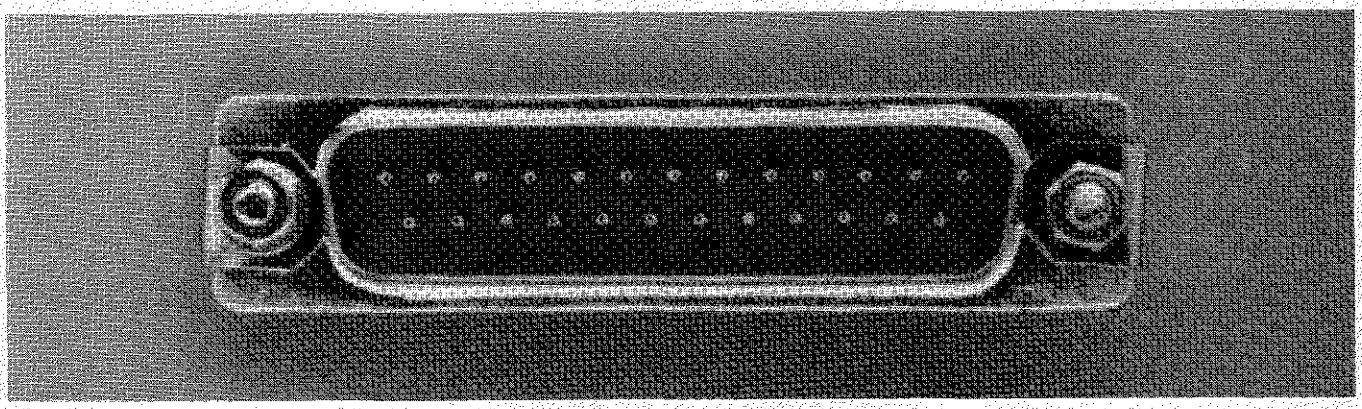
ACUCOBOL-85/the ACUCOBOL logo are trademarks of ACUCOBOL, Incorporated; MS-DOS/XENIX of Microsoft Corporation; NETBIOS/OS/2/AIX of International Business Machines Corporation; VMS/VAX/ULTRIX of Digital Equipment Corporation; UNIX of AT&T Bell Laboratories; RM/COBOL and RM/COBOL-85 of Ryan-McFarland Corporation.

Number 282 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C Programmers

We Know The Ins and Outs of C Communications



South Mountain Software

Programming communications is a time-consuming, complicated, and sometimes tedious programming job. The task of creating a solid communications application can contain more surprises than a magic show.

Professional C programmers have come to rely on South Mountain for fast, safe and efficient communications functions. With a function library as clean and complete as this one, you don't need to know assembler to have complete RS-232 control.

Functions too Numerous to Mention

We can't list all the many functions here, but our library is designed to support the length and breadth of your application's need. From numerous file transfer protocols to intelligent modem control functions, we do it all. If you don't see what you need in the features box below, call us at **800-451-6174**. We probably have what you're looking for.

Freebies

As an added incentive and learning tool, we include the full source code for a BBS system and a terminal package. Both can be incorporated into projects or dissected for programming hints.

We have also designed and included a sub-system for background file transfers. This procedure can save you hours of experimentation and frustration.

BreakOut II Interactive Communications Debugger

For serious communications application development, BreakOut II will save you hours of debugging time. BreakOut II on a PC will allow you to interact with your comm application on a real-time basis, capturing and transmitting data. In the monitor mode (along with a special cabling option) you can monitor the comm session between any two devices on an RS-232 line.

You will be able to spot hardware malfunctions and view line signals up to 19,200 Baud. All of this for hundreds of dollars less than traditional hardware methods. price $125.

Real People, Realistic Policies

Our tech support and sales lines are peopled by real C programmers. They are able and willing to answer your questions before and after your purchase.

We offer 30-day, money-back guarantees on all our products, and never charge a royalty or run-time fee. If you've forgotten what it's like to be treated as an important customer, give us a call today. You'll like the change.

Features

- **Interrupt-driven transmit and receive to 115,200 Baud**
- **Simultaneous operation on up to 34 serial ports**
- **Intelligent modem control functions**
- **XMODEM, YMODEM, YMODEM-g, and KERMIT**
- **XON/XOFF and hardware flow control**
- **Concurrent file transfer sessions in foreground/background**
- **Timer and CTRL-Break handling**
- **Extensive multi-port board support**
- **Both Microsoft C and Turbo C support included**

Essential Communications

Price $249 inc. Source
No royalties, 30-Day Money-Back Guarantee

Other Products

C-Utility, Essential Graphics, Essential B-Tree, ScreenStar, /*resident_C*/, Guido, Hold Everything

1-800-451-6174

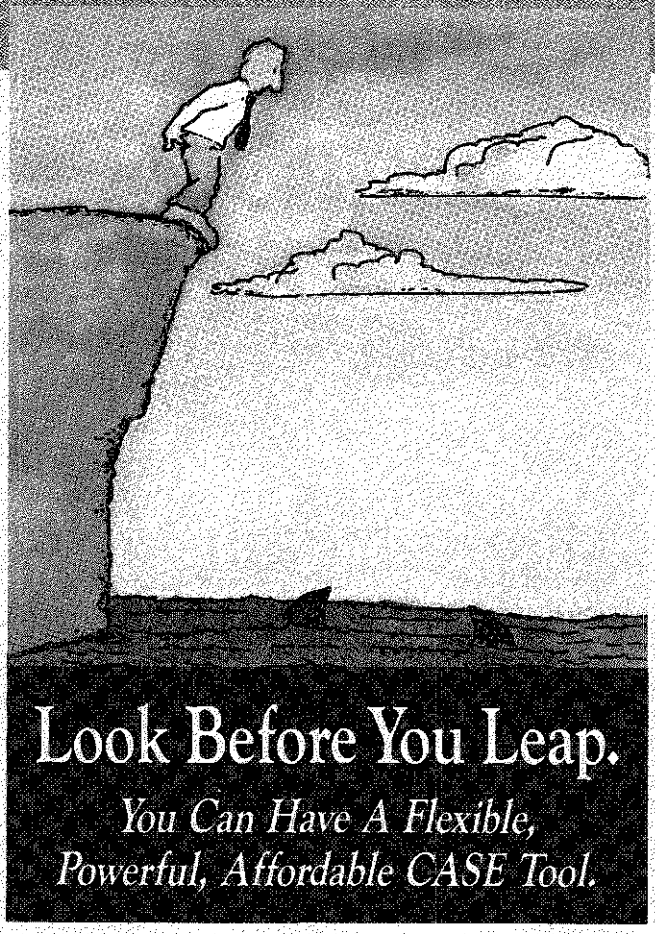
SOUTH MOUNTAIN SOFTWARE™

South Mountain Software Inc.
76 S. Orange Ave., S. Orange, NJ 07079
NJ 201-762-6965 Fax 201-762-0118

Essential Graphics, Essential Communications, Essential B-tree C-Utility Library, Hold Everything, Breakout 2 GUIDO, /* resident_C*/ Screen Star are trademarks of South Mountain Software Inc.

Number **10** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Look Before You Leap.

You Can Have A Flexible, Powerful, Affordable CASE Tool.

Words of Caution and Encouragement for Those Choosing a CASE Tool

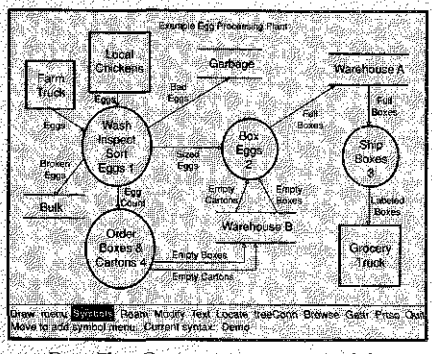
Your software projects are growing more complex, and maintenance is growing more expensive. You think that CASE tools could help, but some say CASE is more trouble than it's worth. What should you believe? Should you jump in? Be cautious. Expensive CASE tools with rigid methodologies demand a commitment of time and money so large that anything short of revolutionary success can seriously damage your budget and your career. Now, however, you can eliminate methodology grid-lock with an affordable, flexible, world-class CASE tool. Literally every programmer and analyst in your company can have the power of vsDesigner™ on their PS/2 or PC.

The Price/Performance Leader

vsDesigner costs only **$495** for a single-user license, but don't be fooled by its low price — vsDesigner has the features and power of CASE tools costing over ten times more. The Expert version of vsDesigner costs $1,995 for a single-user license and includes all vsDesigner features plus LAN support, design database reporting and the ability to modify or create methodologies.

Use Multiple Methodologies

vsDesigner supports the structured design methods you prefer: Yourdon, Gane-Sarson, Warnier-Orr, Ward-Mellor Real-Time, Entity-Relationship, Flow Chart, etc. It supports multiple methodologies or "syntaxes" simultaneously so that a design can contain layers (nodes) using different syntaxes. For example, you can use Data Flow Diagrams for the top level of your design and then decompose each module into Structure Charts, State Diagrams, Entity-Relationship Diagrams, etc.



Data Flow Diagrams are just one of the standard methodologies supported

Customize Methodologies Or Create Your Own

With the Expert version of vsDesigner, you can modify the syntaxes we supply, or you can even create your own syntaxes for ultimate flexibility. We believe that the tool should adapt to and then enforce your team's standards and practices rather than vice versa.

Quality Code From Quality Design

Although many CASE tool users have had significant productivity gains, structured design improves quality even more than productivity. The design checking programs included with vsDesigner allow you to check the logical integrity of your designs before you begin writing code. The resulting code will be more reliable, and maintenance costs will be much lower because operations are logically consistent.

Collaborate over a LAN

The Expert version of vsDesigner also adds LAN support so that all your designers can simultaneously work on the same design. Node locking allows design collaboration while preventing conflicting changes to the shared design.

The Expert version of vsDesigner also provides the database and reporting capabilities necessary to predict lines of code, number of modules, completion dates, responsibilities and project complexity.

One Picture Is Worth A Thousand Meetings

vsDesigner enhances the communication among team members by giving managers, team leaders, programmers and analysts a visual picture of your system's architecture making it faster and easier to communicate design changes throughout the team.

Fast Learning Curve

vsDesigner has powerful on-line help, and the User's Manual has an extensive tutorial. Your choice of Lotus-style menus or pop-up menus make command selection easy. The built-in word processor even emulates popular word processors to reduce learning time. Color graphics make your designs easier to understand.

A World-Class Design/Analysis Tool From The Best-Rated CASE Vendor

When you choose vsDesigner, you become a valued customer of Sage Software, the "Best-Rated CASE Vendor" in the Donohue 1989 Fortune 1000 Software Industry Report. Sage is committed to meeting your needs with excellent customer support, training and best-in-class tools including APS/PC and APS/MVS Application Generators, PVCS, PolyMake and vsDesigner.

Both versions of vsDesigner require an IBM-compatible PC or PS/2 with 640K RAM, EGA monitor and a hard disk. A mouse is recommended. Printers supported are: Epson FX85, FX286 & MX80; HP LaserJet+ and LaserJet Series II.

30-Day Money-Back Guarantee

To Order 1-800-547-4000

Outside North America call (301) 230-3307. Send checks and P.O.s to SAGE/POLYTRON, 1700 NW 167th Place, Beaverton, OR 97006. FAX: (503) 645-4576. Prices are N. American only.

vsDesigner

SAGE SOFTWARE

Number 26 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Customers You've Never Considered

by
William H. Roetzheim

This article is based, with permission, on the author's book Proposal Writing for the Data Processing Consultant *(Prentice-Hall, 1986).*

Several different market segments are available to the independent computer consultant, including:

Local small businesses (for example, doctors, dentists, lawyers, and accountants) often provide the bulk of the day-to-day work.

- Larger national businesses require computer-consulting assistance on either a one-time or an ongoing basis.
- City, county, state, and federal government agencies make frequent use of consultants.
- Prime contractors, such as IBM, Honeywell, and TRW, often hire subcontractors for special computer projects.
- Branches of the military often seek outside expertise on projects.

If all of your clients to date have been local businesses, the thought of marketing to government agencies undoubtedly has you shaking your head. You probably picture the government as some large, gray mass of blank faces and bureaucratic red tape, suitable for penetration only by the largest and most powerful of companies. Boy, are you wrong . . . and I'll prove it to you!

What Is "The Government"

The government is the U.S. Department of Agriculture. The government is the Internal Revenue Service. The government is the Department of Defense. Pretty scary, right? But the government is also your state Parks and Recreation Department, your state Department of Transportation, and your state Employment Development Department—a little better, but not exactly your local mom-and-pop grocery.

OK, what if I said that the government is the library down the street, the local school that your son or daughter attends, and the municipal lake or swimming pool in your area? That's not so bad, is it? My point is that the government is so large, and so diverse, that it includes every type of customer you have ever worked with or imagined working with.

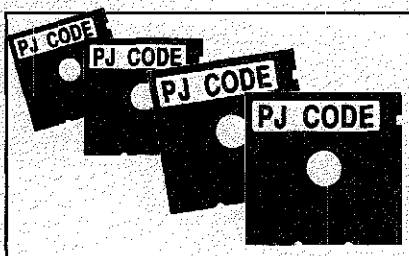
As a computer consultant, if your entire perception of the government were based on contracts with your local county and city government agencies, you would describe the government as being like a small- to medium-sized, well-run, but somewhat rigid, private business. I've worked for these agencies. The working environment is relaxed and informal. The paperwork is minimal. Obtaining the work is so simple it's ridiculous. So, there is absolutely no valid excuse for any computer consultant to avoid marketing to government agencies.

Why the Government Needs Help

It has been my experience that 90 percent of local government work that is funneled to independent computer consultants falls into one of three categories:

Specialized technical assistance typically involves the modification or development of custom programs, training of government personnel in the use of software or hardware, and hardware interfacing. The San Diego Fire Department recently hired a computer consultant to design and develop an application program on a microcomputer to handle the efficient dispatching of ambulances.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



PJ Code Is Available!

On disk from:

Programmer's Journal
P. O. Box 70167
Eugene, OR 97401
800-234-0386
503-747-0800 in Oregon

All Code Disks $10.00 each.
To order, see ad on page 88.

PJ Code DiskScriptions
are available for only $39.95.
To order, see ad on page 8.

On BBS from:

EDI
1332 Hermosa Ave. #7
Hermosa Beach, CA 90254
213-374-3478
Call for your local access number.

Generation Five BBS
(Voice) 301-495-6690
(Data) 301-495-2932
Accessible via PC Pursuit,
Washington D.C. From
Files submenu, select "P"
for Personal Downloads.

Gilmore Systems of Southern California
5447 Indian Hills Drive
Simi Valley, CA 93063
(Voice) 805-582-1360
(Data) 805-582-9306

Ithaca Street BBS
1145 Ithaca
Boulder, CO 80303
(Voice) 303-494-8865
(Data) 303-494-8868

Mnematics Videotex
P.O. Box 19
Sparkill, NY 10976-0019
(Voice) 800-322-3633
(Voice) 914-365-0184 in N.Y.
Mention *PJ* when you call.

Logical Alternatives Incorporated
Calder Square, P.O. Box 10674
State College, PA 16805-0674
(Voice) 814-234-8088
(Data) 814-234-1881

PCS-49 on CompuServe
Section 10
Sponsored by:
MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170
(Voice) 703-450-4181

Later, a different consultant was hired to interface the computer system with an electronic display board showing the current status of all ambulances. Both projects required technical expertise that the fire department did not possess.

Technical quality assurance simply involves monitoring the work of larger computer companies to ensure that the product delivered to the government agency meets the contract's requirements.

Planning and design work may involve hardware-system design, preliminary application-program design, and requirements analysis. This work is often used as a basis for a formally published Request for Proposal (RFP). Both of these areas are almost exclusively limited to small, independent consultants, as most large companies already have ongoing contracts with all branches of the government and would be faced with a potential conflict of interest if they agreed to perform this work. Take advantage of your size and grab these contracts for yourself!

What the Government Requires

If you're just breaking into the computer consulting field, you have probably already found that this is an experience-required job. Local government agencies are one of the few market segments that are normally willing to give your company a chance even if you haven't built up an impressive company resume. What they will require from you is simple:

- Assurance that you can perform the work within the time and money available
- The total absence of any conflict of interest
- A well-written proposal

With a properly written proposal, a small company with little or no experience can get a contract with a local government agency, even if the local competition is ten times greater than expected. Most small companies bidding on local government contracts simply do not know how to write a decent proposal. You can easily use their ignorance to your advantage and thus win the contract.

Advantages of Government Work

I love working for local government agencies for several reasons. The work is a great way to build experience and an impressive company-experience base. The working environment is relaxed and informal. (I once had a small contract with a regional airport in the County of San Diego. The official who hired me met me in my living room on his way to work in the morning!) The paperwork is minimal. They always pay their bills (albeit somewhat slowly). They have realistic expectations about what the computer system or software will do for them and what it will cost.

How to Get Started

At this point, I sincerely hope that I've convinced you to look into this market segment. If you're wondering how to get started, take the following steps:

1. Make an appointment with your local government small-business advisors to discuss the types of work your company is qualified to perform and to solicit advice.
2. Watch the public-announcements section of your local newspaper for solicitations related to computers. The local small-business advisor can tell you in which newspapers the government advertises.
3. Make personal sales calls on local government officials who might use your services. Try to get referrals from one government official to another, leaving each person with a positive impression of you and your company and a clear understanding of the type of work you are qualified to perform.

Take these simple steps and you'll find yourself working on government contracts within six months.

Going after Bigger Fish

As your company grows and becomes more experienced, you may eventually want to try your hand at larger government contracts. The procedures for obtaining state and federal contracts (as well as many subcontracts in this arena) are similar to those for local government. Most states do not advertise

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Software Managers Wanted!

W.H. Roetzheim & Assoc. is currently seeking 100 Beta-testers for version 2.0 of Structured Project Manager's Toolbox (SPMT).

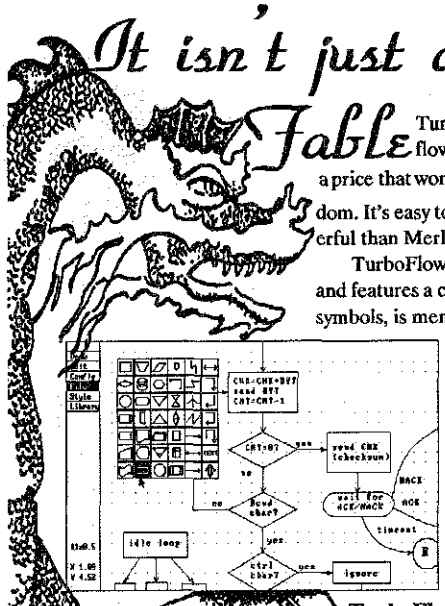
SPMT is a complete project management environment tailored to software development projects. Ease of use without sacrificing power has been our objective. Handles multiple projects in one or many files. Enter up to 10,000 tasks in your WBS, then collapse the WBS to get an overview. Use a mouse to click on WBS elements to make changes. From the Gantt chart, use the mouse to drag tasks to their new start date. Full support for the government's DOD-STD-2167A and C/SCSC cost accounting. Includes LAN support. Site licenses and source code are available. The complete package, including extensive documentation, retails for $492. For a limited time, Beta-testers will receive the entire package, plus free upgrades, for $125. Our usual 30-day money-back guarantee also applies to this special Beta-test offer.

To order, send cash, check or money order — or call and we will ship your Beta-test kit COD. Be sure to mention this ad when ordering.

W.H. Roetzheim & Associates
13518 Jamul Drive, Jamul, CA 92035
(619) 669-6970

Number **339** on the Reader Service Card

It isn't just another Fable



TurboFlow is serious flowcharting software at a price that won't cost you your kingdom. It's easy to use and more powerful than Merlin's magic.

TurboFlow runs on an IBM PC and features a complete set of ANSI symbols, is menu driven, interfaces withdesktop publishing software, and supports a variety of printers and plotters.

So stop living in the dark ages. Call 1-800-882-5822 and order your copy, or ask for our free brochure.

TurboFlow & Logitech serial Mouse $89
TurboFlow $69

Legendary Software from:

Daytron Electronics, Inc.
(214)669-2137
610 S. Sherman, Suite 104, Richardson, Tx 75081

Number **308** on the Reader Service Card

Data Junction® converts data files!

***Any* ASCII • EBCDIC • c-tree • Packed data Binary • dBASE • R:BASE • Paradox • Xdb Excel • 1-2-3 • SC4 • Btrieve • Oracle Informix • Clarion • DIF • SYLK . . . plus many more file types.**

- *No programming — completely menu driven!*
- *Selects & rearranges records, fields and characters!*
- *Instant script file creation for total batch processing!*
- *Maps new data structures automatically by name!*
- *Ideal for OEM/run-time applications!*

". . . an invaluable integration tool." —*Byte*
". . . more than worth its price." —*PC Magazine*
". . . a surefire winner." —*Computer Language*

Standard $99 — dBASE, ASCII, 1-2-3, Q&A, Enable, DIF, askSam & merge files for WordPerfect, Word & Wordstar
Professional $199 — Standard formats plus Binary/EBCDIC, SC4, SYLK Paradox, R:BASE, Clarion, Excel, c-tree & Btrieve
Advanced $299 — Standard & Professional formats plus Informix, Xdb, Oracle & C-ISAM

Call **800/444-1945** or 512/482-0824
MC/VISA • AMEX • COD • Corp. PO.
available for LANs, Xenix & Unix
Tools & Techniques Inc • 1620 W 12 • Austin TX 78703

T&T TOOLS & TECHNIQUES

Number **327** on the Reader Service Card

SOFTWARE DEVELOPERS: ARE YOUR PRODUCTS TOO HARD TO INSTALL?

In today's competitive market, your product **must** be easy to use -- immediately. It is no longer commercially acceptable to tell the end-user to use batch files, COPY, DISKCOPY, RESTORE, or to require wading through paragraphs of text in the manual. The installation process is something that a program developer **should** and now **can easily** completely automate.

INSTALL 2.3 is a bullet-proof automatic installation program you can distribute - royalty free - with your product • Features: Up to 4.3 gigabyte file sizes • Windowing/color user interface • **Full C source** with project, make, link, and batch files • File integrity verification • Intelligent handling of virtually all errors (open disk drive doors, full disks, unformatted disks, etc) • Hard & floppy disk installation • Multiple source & target disks • Multiple target subdirectories (which *INSTALL* will create if necessary) • Unlimited free tech support • Granular installation • Easy method to add-in new functionality • Ability to execute DOS commands • Can intelligently modify CONFIG.SYS & AUTOEXEC.BAT files • Includes **data compressing file librarian** • **NEW** Automatically detects and can base the installation on the CPU (86/286/386), DOS version, display (CGA/B&W/MONO/EGA/VGA), NetBIOS, ANSI.SYS, LIM, etc.

INSTALL offers 3 years of proven reliability that has been used to install over 500,000 copies of products including some of the most prestigious in the industry. We would like to add your name to that list.

ORDER NOW TOLL FREE 1-800-331-2783

KNOWLEDGE DYNAMICS CORPORATION
Highway Contract 4 Box 185-H, Canyon Lake, TX 78133-3508
Voice: **1-512-964-3994** FAX: **1-512-964-3958**
MasterCard/VISA/COD/Purchase Order

$149.95

Number **295** on the Reader Service Card

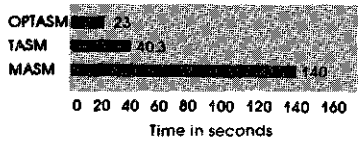
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

THE PROGRAMMER'S CHOICE

OPTASM
- fastest assembler available
- eliminates phase errors
- never inserts spurious NOPs
- complete TSR online help
- includes a linker and debugger
- "OPTASM is clearly technically superior to MASM" *Computer Language - Dec 88*

$125

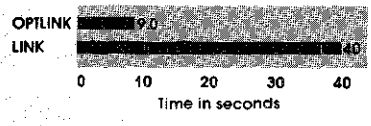
ASSEMBLING **BGIDEMO** ON PC



OPTLINK
- fastest linker available
- generates EXE, COM or SYS
- generates global XREF
- supports CodeView info
- works with MS C5.1

$125

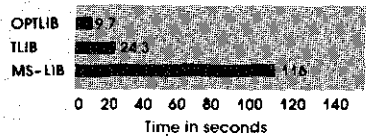
LINKING **BRIEF**



OPTLIB
- 10 times faster than LIB
- * and ? in filenames
- time-based replace
- LIB-compatible syntax
- rename PUBLICs and EXTRNs

$49

BUILD 670K .LIB FILE



OPTDEBUG
- source level - supports CV info
- online help system
- reverse execution
- 50k COM file
- debug EXEs, COMs and TSRs

$125

SLR SYSTEMS
1622 N. Main Street Butler, PA 16001
412-282-0864 TELEX 559215
800-833-3061 (USA)
Call today to order!

Number 165 on the Reader Service Card

procurements in the local newspapers, choosing instead to publish a newspaper-like document containing all announcements of procurements. The federal government uses a similar procedure, calling their publication of pending procurement the *Commerce Business Daily* (or *CBD*). You can review a copy of the *CBD* at many public libraries. When you see a project you are interested in, request a copy of the Request for Proposal (RFP) that describes the work. Then write a proposal describing your qualifications and approach to solving the problem, and send it in to the agency involved. (In a future issue, we will look at the *Commerce Business Daily* in more depth.)

William H. Roetzheim received his M.B.A. in 1983 and is currently an M.S. candidate in computer science. He has worked as senior project manager for Tetra Tech Services (a Honeywell subsidiary) and as a consultant and author specializing in software development for the government. He is the author of several books, including Structured Computer Project Management *(Prentice-Hall, 1988). Mr. Roetzheim has also written an IBM-PC–based project-management tool, Structured Project Manager's Toolbox (SPMT), that is tailored to software-development projects. Mr. Roetzheim may be contacted at 13518 Jamul Drive, Jamul, CA 92035, 619-669-6970.*

Turn your favorite C compiler into a powerful database manager with the

C/Database Toolchest

The **C/Database Toolchest™** adds sophisticated file management functions to your Power C™, Turbo C®, QuickC®, or Microsoft® C compiler. With the **C/Database Toolchest™**, your data requires much less disk space than with programs like dBASE®, and you can access your data much faster. Of course the full power of C provides you with an unlimited amount of programming flexibility.

The **C/Database Toolchest™** includes three major components:

1) An advanced B+tree library gives you instant access to your data.

2) A high-level ISAM library provides you with an easy-to-use C interface, and

3) A complete database manager (with C source code included) shows you how to create impressive applications.

You also receive a comprehensive 350 page manual and a utility for converting dBASE® files.

The **C/Database Toolchest™** supports features that you'd expect to find only in products costing ten times as much. Advanced features include variable length records, variable length keys, multiple keys per index, and multiple indexes stored in a single file. Your data files can contain an unlimited number of records, and each record can be as large as 32K bytes in length.

About the only thing that the **C/Database Toolchest™** doesn't do is cost you a lot of money. We've kept our price low so you can manage your budget as easily as your data.

Now Only $19.95!

Order now by calling our toll free number or mail the coupon to:

Mix Software
1132 Commerce Drive
Richardson, TX 75081

1-800-333-0330

60 Day Money Back Guarantee
Not Copy Protected ■ Royalty Free
For technical support, please call:
1-214-783-6001

MIX software

Order Coupon

Name_____
Street_____
City_____
State_____ Zip_____
Telephone_____
Paying By____ Money Order____Check
____Visa____ MC ____ AX ____ Disc.
Card#_____
Exp. Date_____
Disk Size____ 5¼" ____ 3½"

Qty.	Product	Price	Subtotal
____	C/Database Toolchest	$19.95	____
____	C/Database Library Source. B+tree & ISAM library source code	$10.00	____
	Add Shipping ($5 USA, $20 Foreign)		____
	Texas Residents Add 8% Sales Tax		____
	Total Amount of Your Order		____

C/Database Toolchest and Power C are trademarks of Mix Software. QuickC and Microsoft C are registered trademarks of Microsoft. Turbo C is a registered trademark of Borland. dBASE is a registered trademark of Ashton-Tate.

Number 266 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NOW, advanced overlays and CodeView®!!

NEW PRODUCT

.RTLink™/Plus

•Smaller Memory Images
Multiple and nested overlays - virtually unlimited flexibility

•Reduce .EXE size on disk
.EXEs require less disk space on target machine and shipping media

•Create faster, more efficient programs
Performance profiling at user-adjustable timings to thousandths of a second.

"One month before ship date, PBRUSH.EXE required 420K of disk space and 730K of RAM. After working with .RTLink, we got the program down to 360K of disk space and 600K of RAM. .RTLink let us take a product that was too big, and turn it into our best seller ever..... .RTLink is great - we are now committed to using it on every new DOS product we release."

Mark S. Zachman
President
ZSoft Corporation
PC Paintbrush IV

Linker	Overlays	Hierarchical (nested)overlays	CodeView Support	Run-Time Libraries	Link-Time Profiler	Price
.RTLink/Plus	YES	YES	YES	YES	YES	$495.00
.RTLink	YES	YES	NO	YES	NO	$295.00
Plink86+™	YES	YES	NO	NO	NO	$495.00
MS-Link™	YES	NO	YES	NO	NO	—
TLink™	NO	NO	—	NO	NO	—

Nested Overlays - Hierarchical (nested) overlay management and the ability to overlay data allows smaller memory images than the Microsoft linker's overlay function. 64,000 separate or contiguous locations are allowed within the memory image. Each overlay area can contain smaller, nested overlay areas which can differ depending on which primary overlay is in memory at the time. Code brought into each overlay area (called overlay sections) can be placed into either the executable (.EXE) file or into separate, external files whose names you specify. There can be up to 64,000 (nested) overlay sections. You can specify that a specific overlay section be loaded into a specific overlay area when the program is brought into memory during program startup. You can specify the code which is to be a part of a specific overlay section by naming either object files, modules, or classes. There is no overlay interrupt to conflict with the other interrupts on an end user's system.

Profiling - provides highly detailed performance statistics about your program without code modification. Timing sequences are adjustable to thousandths of a second. Source code is provided for the read/analysis utility which allows customized statistics reporting.

Run-Time Libraries (.RTLs) - .RTLs are optional, easily-created files that can be used to manipulate link speeds, and significantly REDUCE .EXE SIZE ON DISK. Generally, .RTLs are files that contain common code which is repeated in groups of .EXE files. These files can dynamically link to .EXEs at loadtime. They reduce the repetitive storage of common code. This results in smaller .EXEs on disk reducing space requirements on shipping media, target and development machines, and reduces time to modem updates to remote locations. Production version .RTLs usually contain commonly used language library code and/or utility code.

Faster Development Linking - .RTLs are used as a productivity tool during the development and testing phase. All of the code within an .RTL is prelinked. Therefore, you can place stable code into an .RTL to eliminate the need to relink it during development. This use of .RTLs can result in lightning fast linking without affecting program testing. And CodeView is supported!

LIBLIST - crossreferences .OBJ and .LIB files. Illustrates interrelationships and dependencies between modules. Provides much more detailed information than utilities such as .LIB.

OTHER FEATURES:

- Supports overlay caching and extended or expanded memory
- Reloading of overlays on function return
- Removes Microsoft overlay restrictions on function pointers (calls through a pointer)
- External overlays (optional) (store in separate file)
- Quick-start ARF's and help file on disk
- Professional level product support at no additional charge
- No run-time charges or royalties
- No performance degradation
- No source code changes
- Excellent Documentation
- Supports Periscope®
- Soft-ICE™/Magic CV™ support

Order today...

.RTLink/Plus **$495.00**
.RTLink (without CodeView/Profiler) **$295.00**

Pocket Soft, Inc., 7676 Hillmont, Ste. 195
Houston, Texas 77040
(713) 460-5600

For Phone Orders Call
800-456-7032 ***FAX (713) 460-2651***

Authorized Licensee of Microsoft® CodeView® Information. "Microsoft" and "CodeView" are registered trademarks of Microsoft Corporation.

Note; Texas Residents include 7% sales tax

Trademarks: All Pocket Soft Products are trademarks of Pocket Soft, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.
©1989 Pocket Soft Inc.

Number 279 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Circles for the VGA

by
Michael Abrash

Installment 16: In which we learn how to draw circles half-fast.

Whenever I pick up one of the classic books on graphics algorithms, I come away thinking that those guys must be awfully smart, because *I* sure don't understand what they're talking about. The explanations are cryptic and difficult to follow, and they often leave key elements as the legendary "exercise for the reader." Worse, they're difficult to relate to the real world: The authors concentrate on deriving algorithms, when what you really want to know is how the algorithms work and particularly how to implement them efficiently—and that final and most important step is, sadly, often omitted, or is at best shown in marginally useful pseudocode.

I do not know why this state of affairs exists, but exist it does. One theory is that academics think in abstractions, not implementations. Maybe so. Another, more cynical school of thought holds that academics must publish "major" works to get degrees, tenure, grants, and the like, and so they dress up relatively simple concepts with a great deal of theory. Again, maybe so. At any rate, you and I are left holding the bag, with graphics code waiting to be written on the one hand; cryptic, highly theoretical explanations on the other; and a yawning gap in between.

It was against this background that I received a letter from Hal Hardenbergh (famed as the only "first-rate engineer" who writes for the public—see the letter column of the September 1989 *Dr. Dobb's Journal*—and he is indeed first-rate, although not the only such) regarding an article I wrote on line drawing. Hal's letter began:

"Ten pages of typeset text on drawing a LINE??? Yikes! I had in mind 5 pages on drawing three kinds of curves!" (Hal is inordinately fond of italics and exclamation points.)

This gentle missive was followed by a call from Hal asking whether I'd be interested in writing an article with him about drawing circles, ellipses, and tilted ellipses. He felt strongly that there was nothing particularly complicated about any of the above and that one article should do it for all three.

In a world of people who puff themselves up by making the simple complex, I was more than slightly intrigued to meet someone who promised to make the complex simple. Hal and I got together, and while he decided to leave the article writing to me (he merely instructed me in the concepts and gave me his 68000 circle and ellipse drawing code; the man is remarkably generous with his considerable knowledge), he did indeed make the whole topic seem simple. Not quite so simple that it'll fit in one article—I'm going to spend two installments on circle drawing and one each on ellipses and tilted ellipses—but far simpler than the graphics books would make it seem to be, and the credit for the concepts in all four installments goes enthusiastically to Hal.

Hal informs me that he thinks his approach may be the one used by Bresenham's circle-drawing algorithm, but he's not sure; he got fed up with the turgid nature of the standard references and derived his own approach. That's our good fortune, for Hal's approach is both fast—almost in a class with line drawing—and easy to understand.

With the credits and history out of the way, let's get started with circle drawing. We'll spend this installment and the next on circles and then progress to the more flexible but more complicated ellipses and tilted ellipses. In the process, we'll do a good deal of the low-level C and assembly language optimization that we've performed many times before in "On Graphics"; we'll also learn more than a little about the kind of optimization that springs from understanding the operation of an algorithm thoroughly and then reshaping it to match the PC's capabilities. Only by applying both sorts of optimization can we test the limits of the PC's performance.

A Little Background

Circles, ellipses, and tilted ellipses rank no better than fourth on the list of most-used graphics elements, after bitblts, lines, and filled areas. Nonetheless, they are used for many sorts of drawing, particularly in CAD and CAE applications, and their importance is magnified because most software takes so doggoned *long* to draw them.

True circles—that is, objects for which every point drawn is the same distance from the center as measured in pixels (not screen distance)—are useful only on screens that offer square pixels (pixels spaced equally along both axes). On any other sort of screen, true circles come out squashed or distended along one axis

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

or the other. Of the many widely used PC graphics adapters, only one, the VGA, offers square pixels, and then only in modes 11H and 12H. (Actually, the MCGA also supports square pixels in mode 11H, but I've encountered remarkably little developer interest in the MCGA.) Consequently, I'll focus on the VGA in this installment.

The circle-drawing code I'll present will in fact run on the EGA (although the mode and Y-axis range in Listing 2 would have to be changed), but it will produce ovals, not circles, in all EGA modes. Circles on adapters other than the VGA will have to wait for the advent of ellipses—which can be drawn to compensate for nonsquare pixels so that they appear to be circles—two installments from now.

Just to be sure we're all speaking the same language, ellipses are ovals drawn around two foci, with a constant combined distance from the two foci to each point on the ellipse. For our purposes, ellipses will further be defined to have symmetry around the X and Y axes (the foci share either a common Y coordinate or a common X coordinate). Tilted ellipses need not have symmetry around any axis. Basically, ellipses are specific 90-degree alignments of tilted ellipses, and circles are ellipses for which both foci are at the same point. I'll discuss the drawing of circles, ellipses, and tilted ellipses in terms of pixels, not screen distance, except where otherwise noted, because pixels are the fundamental units with which the code we'll develop will work.

For this installment and the next, we'll put ellipses, tilted or otherwise, aside and focus on circles.

The Basics

Drawing a circle is actually very easy. The basic equation for a circle is $Radius^2 = X^2 + Y^2$, which is nothing more than a way of saying that every point on a circle is exactly *Radius* distance away from the center of the circle. (*X* and *Y* form two legs of a right triangle, with *Radius* as the hypotenuse; if you know your Pythagoras, you know how to draw circles.)

Viewing circle drawing in computer-implementation terms, we can first cut our workload by realizing that we actually need to generate only one-eighth of a circle; symmetry does the rest. Next, we can generate that one-eighth-circle arc by starting at the zero setting for the major axis (the axis that advances more rapidly) and the maximum setting (*Radius*) for the minor axis, advancing along the major axis one pixel at a time, and using the formula $MinorAxis = sqrt(Radius^2 - MajorAxis^2)$ to calculate the corresponding minor-axis point, as shown in Figure 1.

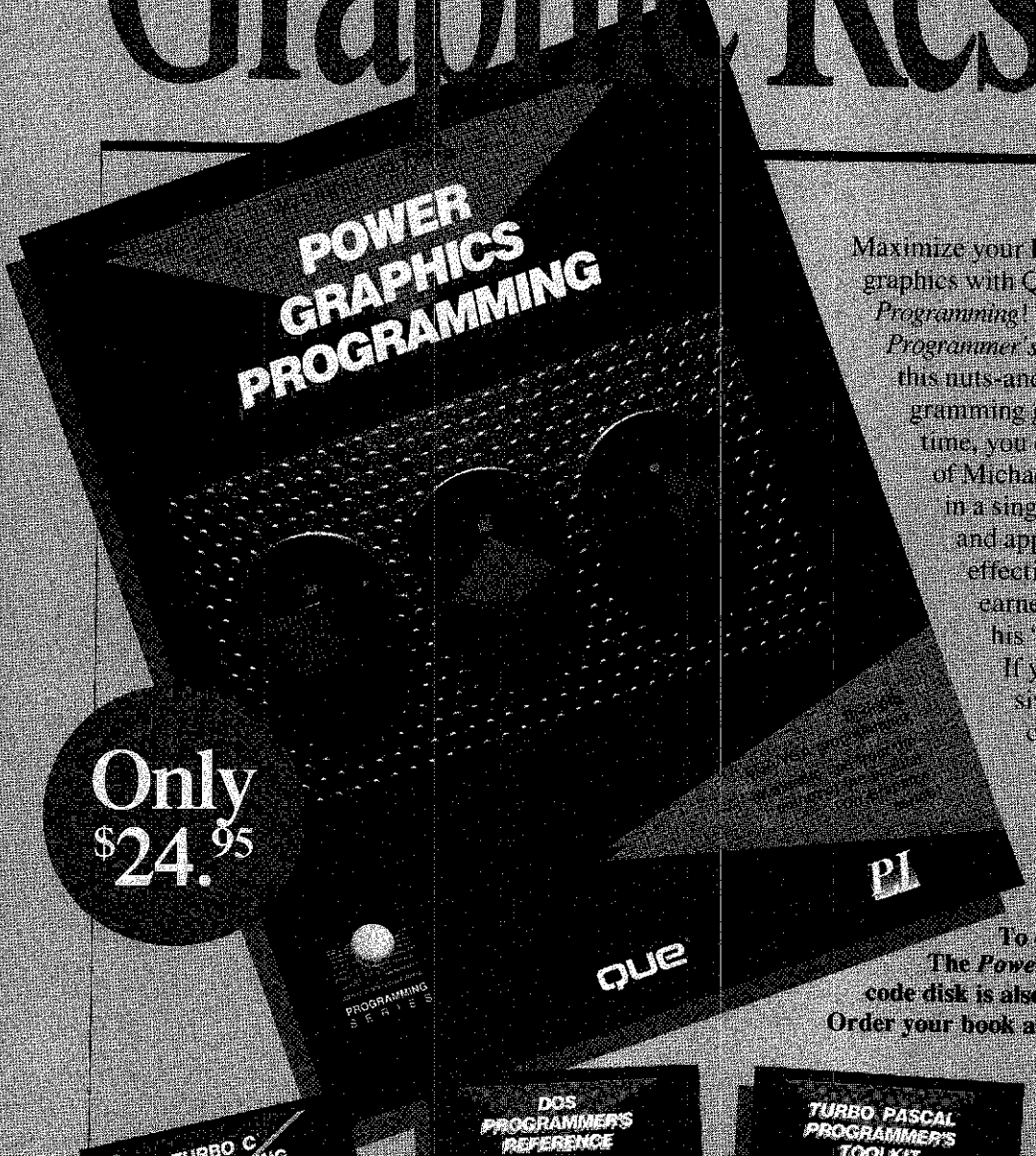
Note that I'm using the terms *major axis* and *minor axis* rather than *X* and *Y*, because the eight-way symmetry allows us to use the same arc calculations when either *X* or *Y* is the major axis. The arc is complete when a point is drawn for which the major-axis coordinate equals or exceeds the minor-axis coordinate. The arc can be regenerated for each of the eight symmetries, but it's easier and less calculation-intensive to draw all eight symmetries for each point at once.

That sounds a little complicated, but it really isn't. Listing 1 shows a straightforward C implementation of the above approach. As you can see, the code is neither long nor particularly complicated, and what complication there is largely results from controlling the VGA's bit-mask and set/reset features. Basically, Listing 1 advances one pixel at a time along the major axis, calculates the corresponding minor-axis point, and then translates that result to all eight symmetries around the center of the circle, drawing the eight points one after another through a dot-plot routine.

Listing 2 is a demonstration program that calls Listing 1 to draw a screen full of multicolored concentric circles. If you run Listing 2, you'll find that it does indeed draw circles, albeit none too quickly. On a 10-MHz AT clone with no floating-point coprocessor, Listing 2 runs in

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Graphic Results!

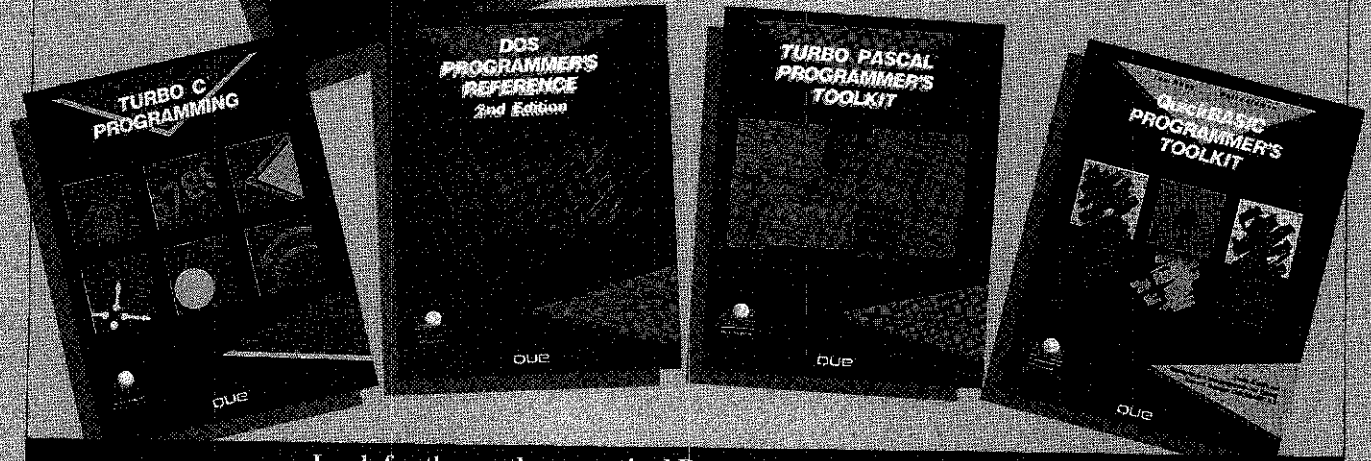


Only $24.95

Maximize your EGA/VGA programming graphics with Que's *Power Graphics Programming!* Que has teamed up with *Programmer's Journal* to bring you this nuts-and-bolts graphics programming guide. Now, for the first time, you can have the expertise of Michael Abrash consolidated in a single volume. His insights and approach to efficient, effective programming have earned him high acclaim for his "On Graphics" series. If you are seeking that elusive graphic edge, you can't be without *Power Graphics Programming!*

ORDER DIRECTLY from *Programmer's Journal* and SAVE!

To order call 800-234-0386. The *Power Graphics Programming* code disk is also available for $14.95. Order your book and code disk today!



Look for these other practical Programming books from Que

Available in better bookstores and computer stores everywhere. Or call 800-428-5331, ext. PJ01.

que®

Number 298 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

17 seconds when linked to Listing 1 and compiled with Turbo C 2.0 and in 27 seconds when compiled with MSC 5.0.

Drawing More Efficiently

Now that we know how to draw a circle, let's think about how we might do it better. There are two possible paths to take: algorithmic refinement and implementation optimization (the latter involving converting to assembly language, code massaging, fine-tuning, and the like). When given this choice, always take the algorithmic approach first, for while it's easy to fine-tune after settling on an algorithm, it's very difficult to change algorithms once you've fine-tuned. We'll concentrate on algorithmic refinement for the remainder of this installment, continuing to work in C code with a dot-plot routine so that the operation of the algorithm is apparent. Then, in the next installment, we'll tighten up the C code by eliminating the independent plotting of each point, and we'll finally arrive at a high-speed assembly language implementation.

The optimization we'll make to Listing 1 is this: Rather than calculating each minor axis point by taking an extremely time-consuming floating-point square root, we'll use integer-only calculations to determine when it's time to advance one pixel along the minor axis. (Granted, a floating-point coprocessor would speed up the square root calculation, but integer approaches are faster still, and most PCs don't have floating-point coprocessors.) In this respect, the faster implementation will be similar to the line-drawing code presented in installment 12 of "On Graphics." Knowing as we do that the minor axis can either advance by only one pixel or not advance at all each time the major axis advances by one pixel, we'll use an integer-threshold variable (which we called an error-term variable in line drawing) to determine when to advance along the minor axis. However, this process is a tad more complicated for circle drawing than for line drawing, because it involves squared values.

Let's take a moment to work out the internals of this new, faster approach. This approach will also beautifully illustrate that you can max out the performance of your code only if you really understand what that code is doing (rather than blindly implementing an algorithm you've looked up), to the point where you can alter the code into forms that are radically different and better suited to the PC but nonetheless functionally equivalent.

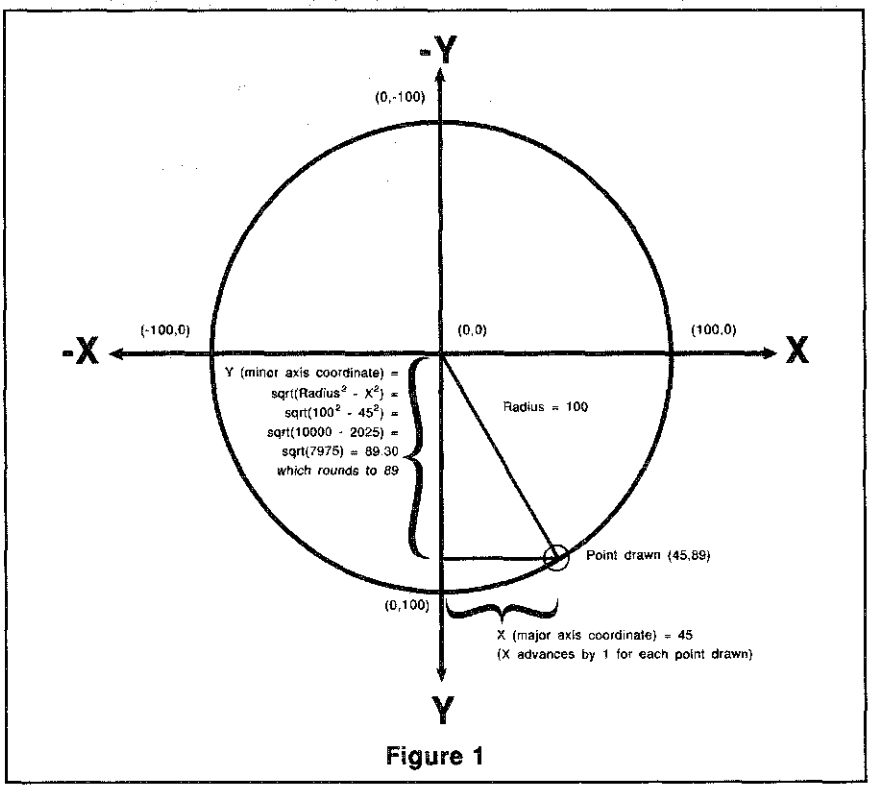


Figure 1

An Incremental Approach

The trick to integer-only circle drawing is to use cumulative expansions of squared elements to keep track of when the minor axis should advance. In other words, instead of calculating each minor-axis point independently as $MinorAxis = sqrt(Radius^2 - MajorAxis^2)$, we'll maintain the running states of $MinorAxis^2$ and $Radius^2 - MajorAxis^2$ and decrement the minor axis by one pixel whenever $Radius^2 - MajorAxis^2$ drops below $MinorAxis^2$.

Okay, I admit it's a *little* more complicated than that, but not much. Consider this: Suppose that to draw a circle of radius 100 centered about (0,0), we start by drawing the arc from (0,100) toward (100,0), as shown in Figure 2. An integer multiply gives us $Radius^2 = 10{,}000$, which also happens to be $Radius^2 - X^2$, because X is 0. Now, as we advance X (the major axis) one pixel at a time, we want to know when it's time to advance Y one pixel; that occurs when Y (which equals $sqrt[Radius^2 - X^2]$) drops below 99.5 (that is, moves more than halfway from the initial Y of 100 to the next pixel along the Y axis, at 99). To put it another way, we want to advance Y when Y^2 (which equals $Radius^2 - X^2$) drops below 99.5^2—or, more conveniently for us, we want to advance Y when $Radius^2 - X^2$ drops below 9900.25.

Why is that convenient? Well, we already know the initial $Radius^2 - X^2$, which is 10,000. It's easy to maintain $Radius^2 - X^2$ as X advances, because $(X + 1)^2$ is simply $X^2 + 2X + 1$, a straightforward integer calculation given X^2 and X. Consequently, $Radius^2 - (X + 1)^2$ can be calculated as $Radius^2 - (X^2 + 2X + 1)$, with no floating-point arithmetic needed.

Now that we know $Radius^2 - X^2$ for each X as we move to the right, all we need to know is the threshold value for $Radius^2 - X^2$ below which we must advance Y. Initially, that threshold value is the value we calculated earlier, 9900.25, which is 99.5^2.

We seem to have a problem here, in that our threshold is a floating-point number. That's easily dealt with, however. First of all, that initial threshold is calculated as $(Y - 0.5)^2$ (the square of the coordinate of

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

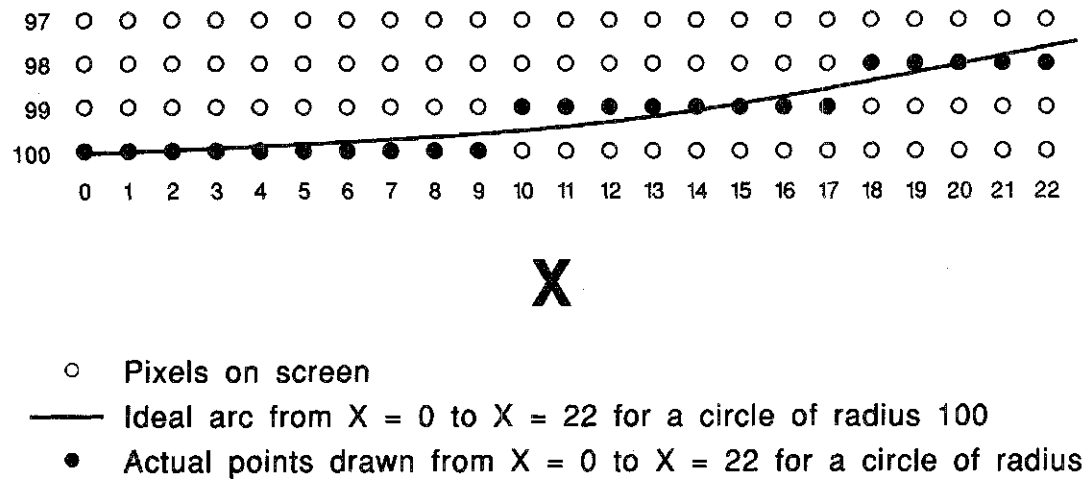

Y
97
98
99
100
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
X
Pixels on screen
Ideal arc from X = 0 to X = 22 for a circle of radius 100
Actual points drawn from X = 0 to X = 22 for a circle of radius 100

Figure 2

the midpoint between Y and Y - 1). That expression expands to Y^2 - Y + 0.25. Now, Y^2 - Y is an easily calculated integer expression; it's only the 0.25 that's a sticking point. It turns out, however, that we can just ignore the 0.25, because the $Radius^2$ - X^2 values we'll compare to the Y thresholds are always integers. If a given $Radius^2$ - X^2 value matches the integer portion of a Y^2 - Y + 0.25 threshold value, we'll know that it actually is less than the threshold, because of the implied 0.25 we'll be carrying along.

For the example in Figure 2, the initial point plotted is (0,100) relative to the center of the circle. (We'll

ProGraphx

ProGraphx Toolbox

Over **50** powerful hi-speed graphics routines for EGA/VGA graphics adapters, including proprietary icon routines to display color images.

ProGraphx Image Editor, a complete color paint package to design color icon images. Edit and view up to 50 icons at once. Simply paint an icon, then use the ProGraphx Toolbox routines to display it with amazing speed. Over **500** sample icons for use in your own programs.

Operation:Airlift, a full screen scrolling arcade style game with source code to demonstrate the power and speed of the ProGraphx Toolbox.

All routines written in assembly for fastest possible performance.

Available for Turbo Pascal, or C.

ProGraphx Menus

Menus and icons without the work!

Mouse driven pull-down menu system. Use a simple text file format to define menu structure along with event codes, then compile and use in application. Only 4 lines of code to add. Event driven menus can be changed and rearranged external to program without recompiling application. ***ProGraphx Menus*** uses the fast routines in ***ProGraphx Toolbox*** to create the fastest response possible. **Powerful, yet easy to implement!**

Also, function-key keyboard menus. Implemented in both text and graphics mode for conformity of applications. Simple to use.

Complete source code for menu-based microcoded processor simulator, along with memory and microcode editor. Provides examples of all windowing routines, in both text and graphics mode.

Available for Turbo Pascal, or C.

ProGraphx Runtime/Source

ProGraphx ToolBox	$199
Complete source	399
ProGraphx Menus	99
Menus Source	199

Specify Turbo Pascal or C version.
*Illinois Residents add 8% sales tax.

ProGraphx
Professional tools with the user in mind.

Send Check or Money Order to:
Stanwood Associates, Inc.
303 East Ohio Street
Chicago, IL 60611

With VISA/MC Call
1-(800) 752-7982
FAX: (312) 828-9742

Number **330** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Develop Powerful Image-based Applications Quickly and Easily

Introducing VICTOR, video capture and image processing library

Support economical frame grabbers

Versions of Victor are available for inexpensive 256 x 256 64-level frame grabbers as well as full-featured boards at 512 x 512 256-level resolution.

Add live video display and image processing features to your application

With Victor your applications can show live video on any portion of the VGA screen and combine live video with text and graphics. Use sophisticated image processing tools, create lineart, TIFF/PCX files, more. Victor handles all the low level functions so you can concentrate on your application.

Combine live video, text, and graphics on the VGA screen.

Rapidly test and prototype

Victor comes with ZIP Image Processing software to get you up and running quickly -- test and prototype functions before you write any code.

Victor supports Microsoft C v 5 and Quick C, and includes over 100 functions, demonstration and prototyping software, full documentation, and source code for device control routines . . . all for only $195.

Use low cost video digitizers to develop high powered applications.

video frame grabbers available

Call (314) 962-7833 to order

VISA/MC/COD

Catenary Systems 470 Belleview St Louis MO 63119 (314) 962-7833

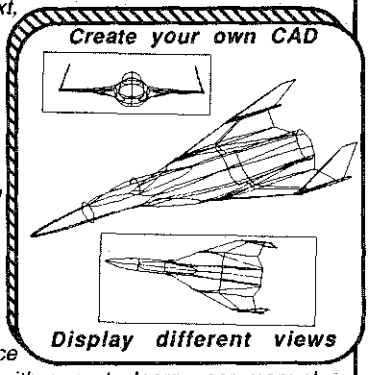
Number **321** on the Reader Service Card

CGL3D New software

A Complete 3D Color Graphic Library for your PC

Create powerful 2D/3D graphic application programs, display 2D/3D spreadsheet and experimental data with functions of an easy to use graphic library.

**2D/3D Point, Line, Text, Polygon, Fill, Surface*
**World coordinates, multiple viewports*
**True 3D perspective*
**Text editor*
**Element color/type*
**Zoom, Transformation*
**2D/3D Views*
**2D/3D Rotation*
**2D/3D Conversion*
**PC MOUSE interface*
**Plot 3D surface data*
**IBM ROM BIOS interface*



**Example source code with easy to learn user manual*
**The Library functions can be access by FORTRAN or C programs. Library is written in FORTRAN 77 and compiled with Microsoft FORTRAN compiler V4.*

Similar in most functions to Tektronix PLOT-10(TCS) mainframe graphic plus additional true 3D functions

To order, Please send $175 plus $3 for shipping to:

F.G.SOFT

P.O. BOX 2152,
Hawthorne,CA 90251-2152

Get user manual and demo flopy disk for $10, refundable upon purchase

Tel 213-542-5461

Number **307** on the Reader Service Card

34010 Video Co-Processors from PC Tech

- Mono 2048 x 1536 displayed pixels
- Color 1024 x 768 displayed pixels
- Up to 5 M byte memory
- Optional CCITT imaging, TIGA, & DGIS software drivers
- C Compiler, Assembler, Linker, Debugger, DGIS Development Kit & TIGA Toolkit available
- PC/XT/AT compatible
- Custom designs available
- **PC Tech understands the 34010**

For further information, please contact:

PC Tech

907 N. 6th St.
Lake City, MN 55041
(612) 345-4555 (Voice)
(612) 345-5514 (FAX)

PC Tech is a trademark of PC Tech, Inc.

PC, XT, AT, DGIS, and TIGA are trademarks or registered trademarks of their respective companies.

Number **208** on the Reader Service Card

PRINTER GRAPHICS IN HIGH RESOLUTION!

GRAPHLINK™ Printer Graphics Toolkit

GraphLink™ lets your programs tap the *full potential* of printers to produce dazzling *printed* graphics in *high-resolution!* Compare the features ...

- Uses the PRINTER'S resolution, not the SCREEN'S!
- Can produce 300 dpi images on laser printers
- A linkable library of 80+ routines for building and printing images
- Provides line, bar, arc, pie-slice, pattern, and bitblt routines, fully-scalable fonts and much more!
- Works with any screen graphics software
- Emulates the Borland Graphics Interface (great for BGI programs!)
- Can *automatically* build the printed image, by intercepting your screen graphics calls (Turbo Pascal version)
- Supports the most popular laser and dot matrix printers
- Supports Turbo Pascal 4.0 and higher. Available soon for Turbo C, Microsoft C, and Quick C
- Only *$125* for the single-user version, or *$250* for GraphLink *Professional*, which includes an unlimited distribution license. (Shipping & Handling: USA $5, Canada $10, Other $15.)

30 Day No Risk Money-Back Guarantee

GRAPHLINK
The Printer Graphics Solution™

VISITECH SOFTWARE
D5 3807 Ridgewood Court
Pittsburgh, PA 15239 USA
(412) 733-4775

Number **304** on the Reader Service Card

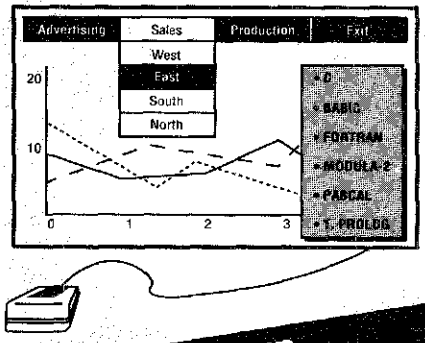
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

IF YOU PROGRAM, PLEASE TRY

HI-SCREEN XL™

WINDOWS - MENUS
DATA ENTRY - HELP

- **WYSIWYG** screen editor • draw • paint • copy • merge • undo • on-line ASCII table
- **WINDOWS** • overlapping • scrollable • for menus, data entry, help screens
- **MENUS** • pop-up • pull-down • scrolling • Lotus-style • "your-own-style"
- **HELP** • context-sensitive • at screen and field level • fully automated*
- **DATA VALIDATION** • type • range • format • color • justification • case
- **INPUT MANAGEMENT** • field-by-field and full-screen input • user-defined exit & function keys • arrays • hot keys



WITH MOUSE AND
GRAPHICS SUPPORT*

- Use the **mouse** to select, scroll, validate, move cursor, click in dialog boxes, etc.
- Mix **graphics and text** screens in your programs. Capture any CGA, EGA or VGA graphic images and combine them with menus, windows, dialog boxes, etc.

Call toll-free: **1-800-338-2852**
in CA: 415-397-4666 • Fax: 415-397-4677

HI-SCREEN XL.....$149
HI-SCREEN XL *Professional Series*.....$325

30 day money-back guarantee • No royalties
* Professional Series version only

For C, BASIC, FORTRAN
COBOL, dBASE, PASCAL...

Softway, Inc.
500 Sutter #222, San Francisco, CA 94102

Number 110 on the Reader Service Card

always do our calculations relative to the center of the circle, then adjust for the location of the center of the circle and perform symmetry calculations later.) $Radius^2 - X^2$ is 10,000 at this point; the initial threshold, $(Y - 0.5)^2$, is 9900.25, but we'll use $Y^2 - Y$ to derive a working threshold of 9900.

Next, X advances one pixel to coordinate 1. $Radius^2 - X^2$ goes to 10,000 - (2X + 1), which is 9999 (X refers to the last X drawn, which is 0 at the moment.) That still exceeds the threshold of 9900, and so Y doesn't change. X then advances to coordinate 2. $Radius^2 - X^2$ goes to 9999 - (2X + 1), which is 9996; Y still doesn't advance. $Radius^2 - X^2$ then advances to 9991, 9984, 9975, 9964, 9951, 9936, 9919, and finally to 9900 as X advances to 10. Remember, we've performed only integer addition and subtraction as X has moved, and a mere two integer multiplications were required to set up the squared terms at the outset.

When X equals 10, $Radius^2 - X^2$ finally matches our threshold for advancing Y; in fact, because of the implied 0.25 we've been carrying around, we know that $Radius^2 - X^2$ has passed the threshold. Consequently, we subtract 1 from Y to advance to the next pixel along the Y axis and then draw the next point at (10,99). That's precisely what we want, since the rounded integer portion of $sqrt(100^2 - 10^2)$ (that is, $sqrt[Radius^2 - X^2]$) is 99.

Our only remaining task is to reset the threshold so that we know when to advance Y again. That's accomplished by calculating the new threshold as $(Y - 1)^2$; much as we did earlier with X, we can expand this to $Y^2 - 2Y + 1$ to avoid multiplication. Given that we already know the Y^2 value for the threshold we just reached, it's a simple matter to calculate $(Y - 1)^2$ for the next threshold each time we advance Y.

There's a trick here, though: The Y we use in the threshold isn't the Y we just advanced to *or* the Y we just advanced from—it's the Y of the threshold we just reached: 99.5 in our sample case. Consequently, when Y advances to 99, the threshold advances to 9900 - (99.5 * 2) + 1, which is 9702. The 0.5 element doesn't mean we have to perform floating-point arithmetic, though, because 0.5 * 2 is always 1. So, the above equation can also be expressed as 9900 - (99 * 2 + 1) + 1, or 9900 - 99 * 2 - 1 + 1, which reduces to 9900 - 99 * 2. Consequently, the formula we'll use to advance the threshold is $Y^2 - 2Y$, where Y^2 is the last threshold value and Y is the Y coordinate just advanced to.

That's really all there is to drawing circles. Reread the last section and work through an example or two of your own, and you'll see that there's nothing to it. It's remarkably easy to implement this approach, too—Listing 3, which is such an implementation, is scarcely longer than Listing 1. Listing 3 is functionally equivalent to Listing 1, except that when linked to Listing 2, Listing 3 runs in 3 seconds on a 10-MHz AT,—*more than five times faster than Listing 1.*

Integer arithmetic is a wonderful thing, isn't it?

Implementation Notes

The obvious question about Listing 3 is "Does it work?" It's certainly fast, but if it doesn't draw circles properly, it's of little use.

I'm confident that Listing 3 does indeed work. The theory is sound; also, I compared the coordinates generated by Listing 1 when linked to Listing 2 to the coordinates generated by Listing 3 when linked to Listing 2, and the results were the same for all points of all circles drawn.

Listing 3 uses long integers, an unavoidable consequence of lugging around squared terms of values that exceed 255. Long arithmetic is relatively slow; it might be worthwhile to special-case circles of radius 255 and less and draw them with a separate routine that uses short integers.

Listings 1 and 3 treat the Y coordinates of circle centers as increasing from the top of the screen to the bottom, contrary to normal graphing convention. If necessary, the sense of the Y axis can be inverted by subtracting the Y coordinate of the center of each circle from the screen height minus 1.

Finally, note that dots are drawn with the statement ***ScreenPtr |= 0xFF;**. The detailed mechanics of

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

why this works—the operation of the bit-mask and set/reset circuitry—have been discussed in past installments, and I can't spare the space to explain them again. An important point, however, is that it shouldn't matter what value is ORed with display memory, because set/reset is enabled for all planes. With Microsoft C 5.0 it *does* matter, though; the problem is that if zero is ORed with display memory, the compiler concludes that the operation will have no effect and optimizes away the entire instruction! Of course, the operation does have an effect, but the compiler has no way of knowing that we're accessing VGA memory, not normal system memory, and ORing with zero does indeed have no effect when accessing system memory. The solution is obvious: OR a nonzero value with display memory.

In Conclusion

That takes us halfway through our exploration of circle drawing. Coming up in the next installment is the other portion of our optimization agenda: fine-tuning for the VGA and conversion to assembly language. The result should be an additional improvement of at least two times (bringing the total increase to at least ten times over Listing 1), but I haven't written the code yet, so we'll just have to wait and see. We have our schedule pretty well laid out for the next few installments, but, as always, I welcome questions and suggestions.

See you next time. PJ

Michael Abrash is a senior engineer working on high-performance graphics software at Metagraphics, in Scotts Valley, California, and a contributing editor of Programmer's Journal. *Michael's book* Power Graphics Programming, *an anthology of his "On Graphics" articles, published by Que Corporation and* PJ, *is available in bookstores and by mail order. Michael is also the author of* Zen of Assembly Language, *published by Scott, Foresman & Company. Readers may write to Michael at 1599 Bittern Drive, Sunnyvale, CA 94087, or contact him via MCI Mail, user name* MABRASH.

Code follows.

GRAPHICS!

Add lightning fast graphics to your programs quickly and easily through the popular PCX file format. Why reinvent the wheel? Make your programs immediately compatible with hundreds of packages from Aldus PageMaker to ZSoft's PC Paintbrush with these linkable graphic libraries.

"An exceptional product" - Programmer's Journal, Aug
NEW! Version 3.5 of the **PCX Programmer's Toolkit** gives you over 60 powerful functions to manipulate bitmapped graphics. Use Virtual screens, Super VGA modes, LIM 4.0 support, a 300 page manual, 9 utilities including screen capture and display, and the fastest routines on the market. **$195**

Need Special Effects, but caught in a GRASP?
Why create a demo when you can create the real thing? Don't be trapped in a slideshow editor or demo program when you can use **PCX Effects** for the PCX Toolkit and your favorite programming language. A Music Language and spectacular effects for exploding your graphics! **$99**

Blazing Graphics Text
With **PCX Text** you can display text with graphics as fast as it always should have been. Display characters, strings, fixed and proportional text, background transparency, and more. Includes a font editor, 85 fonts, and text utilities for blazing graphics bitmapped text. **$149**

All packages support 12 compilers for C, Pascal, Basic, Fortran, Assembly, and Clipper. All modes of the Hercules, CGA, EGA, VGA, and Super VGA adapters are supported, up through 800x600x256 (22 modes in all). Assembly Language source code is optionally available. Trademarks are property of their respective holders.

No Royalties! 30-day Money Back Guarantee.
VISA/MC/AMEX/COD/PO accepted.

G·E·N·U·S MICROPROGRAMMING
11315 Meadow Lake • Houston, Texas 77077 • (713) 870-0737

1-800-227-0918

Number **261** on the Reader Service Card

"The total cost involved in writing one TurboGeometry Library routine is more than the cost of the library".

TurboGeometry Library

Are you programming or planning to program CAD/CAM or graphics applications? Many hours even days can be spent in writing and debugging geometric routines. *TurboGeometry Library* can relieve you of those tasks that are part of every CAD/CAM or graphics program. *TurboGeometry Library* is a toolbox of device independent 2 & 3 dimensional geometric routines that gives you complete flexibility in designing applications. There are over 300 routines in the library. The previous version had:

- 2D & 3D Intersections.
- 2D & 3D transformations.
- Decomposition of polygons
- Hidden line removable.
- Volume & area calculations.
- Perspectives.
- Polygon and line clipping
- and many more.

The new version 3.0 adds:

- Surfacing
- Spherical Geometry
- Solid Geometry
- Tangents
- More curve routines
- 3D clipping
- Link list library
- and more.

TurboGeometry Library comes with a manual, example programs and full source code. 30 day money back guarantee. Needs IBM PC/ Comp, MS/PC DOS. Turbo Pascal, Turbo C, MCS, Power C, Zortech C++. The suggested retail price, for TurboGeometry Library 3.0, is $199.95US. Foreign orders $225.00US. Both prices include shipping and handling. Texas residents add 8% tax. VISA, MasterCard, Chk, PO, MO accepted.

To order Toll Free call **1-800-635-7760**. *To request information call* **214-423-7288** or write to:

Disk Software, Inc.
2116 E. Arapaho Rd. #487
Richardson, TX USA 75081

"In CAD/CAM or graphics it all comes down to using geometry"

FAX 214-423-4465

Number **229** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 88.

Code from "Circles for the VGA," by Michael Abrash

1

LISTING 1

```
/* Draws a circle of the specified radius and color.
 * Compiles with either Turbo C 2.0 or MSC 5.0.
 * Will work on VGA or EGA, but will draw what appears to be an
 * ellipse in non-square-pixel modes.
 */

#include <math.h>
#include <dos.h>

/* Handle differences between Turbo C & MSC. Note that Turbo C accepts
   outp as a synonym for outportb, but not outpw for outport */
#ifdef __TURBOC__
#define outpw  outport
#endif

#define SCREEN_WIDTH_IN_BYTES 80     /* # of bytes across one scan
                                        line in mode 12h */
#define SCREEN_SEGMENT        0xA000 /* mode 12h display memory seg */
#define GC_INDEX              0x3CE /* Graphics Controller port */
#define SET_RESET_INDEX       0      /* Set/Reset reg index in GC */
#define SET_RESET_ENABLE_INDEX 1     /* Set/Reset Enable reg index
                                        in GC */
#define BIT_MASK_INDEX        8      /* Bit Mask reg index in GC */

/* Draws a pixel at screen coordinate (X,Y) */
void DrawDot(int X, int Y) {
   unsigned char far *ScreenPtr;

   /* Point to the byte the pixel is in */
#ifdef __TURBOC__
   ScreenPtr = MK_FP(SCREEN_SEGMENT,
      (Y * SCREEN_WIDTH_IN_BYTES) + (X / 8));
#else
   FP_SEG(ScreenPtr) = SCREEN_SEGMENT;
   FP_OFF(ScreenPtr) =(Y * SCREEN_WIDTH_IN_BYTES) + (X / 8);
#endif

   /* Set the bit mask within the byte for the pixel */
   outp(GC_INDEX + 1, 0x80 >> (X & 0x07));

   /* Draw the pixel. ORed to force read/write to load latches.
      Data written doesn't matter, because set/reset is enabled
      for all planes. Note: don't OR with 0; MSC optimizes that
      statement to no operation */
   *ScreenPtr |= 0xFF;
}

/* Draws a circle of radius Radius in color Color centered at
 * screen coordinate (X,Y) */
void DrawCircle(int X, int Y, int Radius, int Color) {
   int MajorAxis, MinorAxis;
   double RadiusSquared = (double) Radius * Radius;
```

2

```
   /* Set drawing color via set/reset */
   outpw(GC_INDEX, (0x0F << 8) | SET_RESET_ENABLE_INDEX);
                              /* enable set/reset for all planes */
   outpw(GC_INDEX, (Color << 8) | SET_RESET_INDEX);
                              /* set set/reset (drawing) color */
   outp(GC_INDEX, BIT_MASK_INDEX);
                           /* leave the GC Index reg pointing to
                              the Bit Mask reg */

   /* Set up to draw the circle by setting the initial point to one
      end of the 1/8th of a circle arc we'll draw */
   MajorAxis = 0;
   MinorAxis = Radius;

   /* Draw all points along an arc of 1/8th of the circle, drawing
      all 8 symmetries at the same time */
   do {
      /* Draw all 8 symmetries of current point */
      DrawDot(X+MajorAxis, Y-MinorAxis);
      DrawDot(X-MajorAxis, Y-MinorAxis);
      DrawDot(X+MajorAxis, Y+MinorAxis);
      DrawDot(X-MajorAxis, Y+MinorAxis);
      DrawDot(X+MinorAxis, Y-MajorAxis);
      DrawDot(X-MinorAxis, Y-MajorAxis);
      DrawDot(X+MinorAxis, Y+MajorAxis);
      DrawDot(X-MinorAxis, Y+MajorAxis);
      MajorAxis++;          /* advance one pixel along the major axis */
      MinorAxis =
         sqrt(RadiusSquared - ((double) MajorAxis * MajorAxis)) + 0.5;
                           /* calculate the corresponding point along
                              the minor axis */
   } while ( MajorAxis <= MinorAxis );

   /* Reset the Bit Mask register to normal */
   outp(GC_INDEX + 1, 0xFF);

   /* Turn off set/reset enable */
   outpw(GC_INDEX, (0x00 << 8) | SET_RESET_ENABLE_INDEX);
}
```

LISTING 2

```
/* Draws a series of concentric circles.
 * For VGA only, because mode 12h is unique to VGA.
 * Compile and link with listingX.c (where X is 1 or 3) with:
 *    tcc listingX listing2      (Turbo C)
 *    cl listingX.c listing2.c   (MSC)
 */

#include <dos.h>

main() {
   int Radius, Temp, Color;
   union REGS Regs;

   /* Select VGA's hi-res 640x480 graphics mode, mode 12h */
   Regs.x.ax = 0x0012;
   int86(0x10, &Regs, &Regs);
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
    /* Draw concentric circles */
    for ( Radius = 10, Color = 7; Radius < 240; Radius += 2 ) {
        DrawCircle(640/2, 480/2, Radius, Color);
        Color = (Color + 1) & 0x0F;   /* cycle through 16 colors */
    }

    /* Wait for a key, restore text mode, and done */
    scanf("%c", &Temp);
    Regs.x.ax = 0x0003;
    int86(0x10, &Regs, &Regs);
}
```

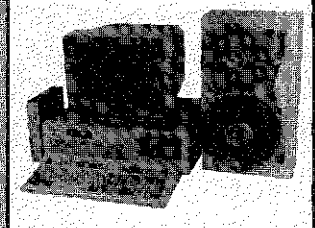
LISTING 3

```
/* Draws a circle of the specified radius and color, using a fast
 * integer-only & square-root-free approach.
 * Compiles with either Turbo C 2.0 or MSC 5.0.
 * Will work on VGA or EGA, but will draw what appears to be an
 * ellipse in non-square-pixel modes.
 */

#include <dos.h>

/* Handle differences between Turbo C & MSC. Note that Turbo C accepts
   outp as a synonym for outportb, but not outpw for outport */
#ifdef __TURBOC__
#define outpw  outport
#endif

#define SCREEN_WIDTH_IN_BYTES 80    /* # of bytes across one scan
                                       line in mode 12h */
#define SCREEN_SEGMENT        0xA000 /* mode 12h display memory seg */
#define GC_INDEX              0x3CE /* Graphics Controller port */
#define SET_RESET_INDEX       0     /* Set/Reset reg index in GC */
#define SET_RESET_ENABLE_INDEX 1    /* Set/Reset Enable reg index
                                       in GC */
#define BIT_MASK_INDEX        8     /* Bit Mask reg index in GC */

/* Draws a pixel at screen coordinate (X,Y) */
void DrawDot(int X, int Y) {
    unsigned char far *ScreenPtr;

    /* Point to the byte the pixel is in */
#ifdef __TURBOC__
    ScreenPtr = MK_FP(SCREEN_SEGMENT,
                      (Y * SCREEN_WIDTH_IN_BYTES) + (X / 8));
#else
    FP_SEG(ScreenPtr) = SCREEN_SEGMENT;
    FP_OFF(ScreenPtr) = (Y * SCREEN_WIDTH_IN_BYTES) + (X / 8);
#endif

    /* Set the bit mask within the byte for the pixel */
    outp(GC_INDEX + 1, 0x80 >> (X & 0x07));

    /* Draw the pixel. ORed to force read/write to load latches.
       Data written doesn't matter, because set/reset is enabled
       for all planes. Note: don't OR with 0; MSC optimizes that
```

Code continues

Desktop 9-Track Tape Subsystem

Now, 9-track tape lets your micro exchange data with minis and mainframes



Simply exchange data files on a reel of 9-track tape.

9-TRACK is the first choice for file interchange among data processing professionals. Now, Qualstar's low cost 1/2-inch 9-track Streaming tape systems bring full ANSI data interchange to IBM PCs or Macintosh, giving your micro the freedom to exchange data files with nearly any mainframe or minicomputer in the world.

Available in both 7" and 10-1/2" versions, compact Qualstar tape drives can sit on your desktop, *using less space than an ordinary sheet of paper.* Systems include DOS or XENIX compatible software, coupler card and cables. High reliability 1600 or 6250 BPI capability may be used for disk backup as well as data interchange. Discover the big advantage 9-track tape has over other micro/mainframe links.

DEC HP ALTOS NCR — IBM MAINFRAMES — AT&T UNISYS APOLLO SUN

Call us today!

FOR DETAILS AND TO ORDER
FAX (818) 882-4081
PHONE (818) 882-5822

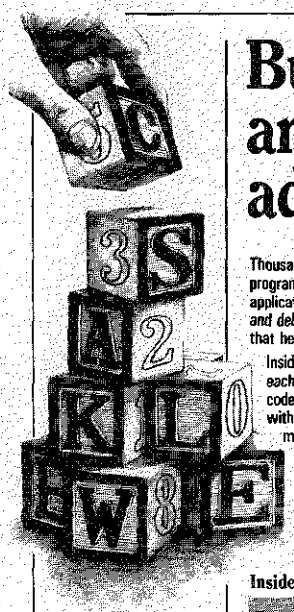
#1 Selling 9-Track Systems on the Desktop

QUALSTAR

9621 Irondale Ave., Chatsworth, CA 91311

©1989 Qualstar Corp. All product and company names and trademarks are the exclusive property of their respective owners.

Number 97 on the Reader Service Card



Build with an Inside! advantage...

Thousands of C, BASIC, FORTRAN and Modula-2 programmers are already using Inside! to improve application performance. You have the best compiler and debugger, *now add the performance analysis tool* that helps create world class applications!

Inside! analyzes your program in real-time, timing *each instance of a function, source line or arbitrary* code fragment, all with microsecond accuracy and without source code modification. Utilize the following modes to know where to focus your optimization efforts for maximum reward.

- Function/procedure timing
- Call target timing
- Source Line timing
- Arbitrary event timing
- Non-executed functions or source lines
- MS-DOS function timing
- Overlay and assembly language support

Inside! is available for popular compilers . . .

- Turbo C
- Turbo Pascal
- Microsoft C
- Microsoft QuickC
- Microsoft Pascal
- Microsoft FORTRAN
- Microsoft QuickBASIC
- Microsoft Modula-2
- Microsoft Macro Assembler
- VAX/VMS C
- Compaq Pascal

Call or write today for a free brochure and the latest list of Inside! products.

To Place Orders
(800)537-5043
Product Support
(508)478-0499
Visa/Mastercard/C.O.D. Accepted

$125.00 EACH
Satisfaction guaranteed

Paradigm Systems
P.O. Box 152 Milford, Massachusetts 01757

Inside! is a trademark of Paradigm Systems.

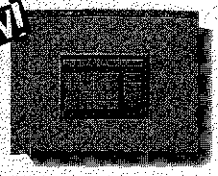
Contact this advertiser directly.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Our Family.

Menuet

NEW!



The most sophisticated and intuitive Graphical User Interface (GUI) for the DOS environment. Object oriented. Fully extensible. Menuet provides over 350 callable functions. **$250 Call for your free Demo!**

- Professional-quality GUI for DOS
- Movable and resizable windows, pull-down menus, dialogue boxes, text entry forms, radio buttons, scrollable lists, signal boxes and gauges, scrollable displays, rubber-banding functions, and more!
- Includes the Interface Design Tool™ (IDT) for prototyping applications and automatic source code generation

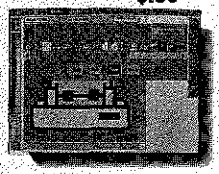
PiXelPrint

NEW!

A utility for both developers and end users, PiXelPrint is based on our innovative Baby Driver printer technology and allows you to print a wide range of graphical file formats to *any* monochrome or color printer we support. **$75**

- Prints virtually any Paintbrush .PCX, GEM .IMG, Microsoft .MSP, Word Perfect .WPG, First Publisher .ART, or TIFF .TIF file to any printer. Color or monochrome.
- Intelligently maps 2, 16, or 256 color images to 2, 8, or 16 color printers. For example, the illustration to the left shows a 256-color 640-480 .PCX file printed on a HP LaserJet II. The exact same file could print to a PaintJet in 16 colors.

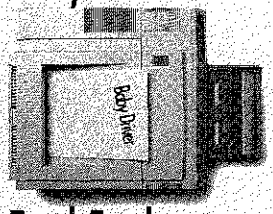
Icon-Tools



Our powerful Icon creation software. Combines all the best features of leading font and paint programs into one intuitive, easy-to-use package. **$150**

- 2,4,8, or 16 colors per icon.
- Up to 32000 icons per file
- Icons as large as the screen
- Editor with dozens of functions, read PCX, MSP, IMG files, colorize B&W images, resize icons and much more

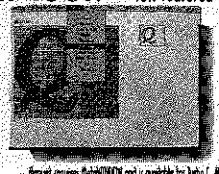
Baby Driver



If you need integrated printer support for your graphics applications, Baby Driver is virtually the only product of its kind in the marketplace. Implemented as a linkable function library, not a TSR, your program controls what, where, when, and how graphics images are printed. **$250**

- Support for over 200 dot matrix, inkjet, b/w and color laser printers
- Prints images in true color on printers like the HP PaintJet and the Tektronix 4696, or renders colors as gray scales on monochrome printers
- Scales and rotates, and positions images
- Includes PCL support for downloading, positioning, and sizing HP soft fonts
- Print from off-screen in-memory images
- Compact function library links directly with your application. No TSRs.
- Device Table editor and Install utility

Font-Tools



Complete utility for creating and using fonts in graphics modes. Includes full-featured font editor with an incredible array of great effects! **$150**

- Font support for DOS graphics programs that can be used with any product
- Font editor for creating and editing fonts. Italic, Bold, 3-D, and many more
- Utilities for modifying, viewing, downloading, and converting soft fonts
- Complete function library
- Over 75 sample font and icon files

Menuet requires Flash/WIDGN and is available for Turbo C, Microsoft C, Zortech C, and C++. Source available for Menuet, Font-Tools, and Icon-Tools. Call for pricing.

303-494-8865

ITHACA STREET SOFTWARE

1445 ITHACA DRIVE, BOULDER, CO. 80303. USA TM

FAX: (303) 494-7802 BBS: (303) 494-8868

Number **226** on the Reader Service Card

Total control with *LMI FORTH*™

For Programming Professionals: an expanding family of compatible, high-performance compilers for microcomputers

For Development:
Interactive Forth-83 Interpreter/Compilers for MS-DOS, OS/2, and the 80386

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 500 page manual written in plain English
- Support for graphics,floating point, native code generation

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, 6303, 6809, 68HC11,34010, V25, RTX-2000
- No license fee or royalty for compiled applications

Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone Credit Card Orders to: (213) 306-7412
FAX: (213) 301-0761

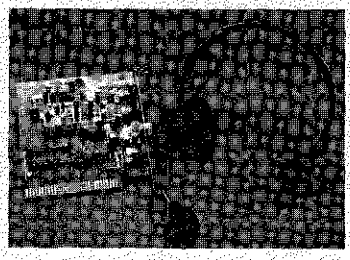
Number **63** on the Reader Service Card

VOICE MASTER KEY VOICE RECOGNITION SYSTEM

FOR PC/COMPATIBLES & TANDY 1000 SERIES A FULL FEATURED VOICE I/O SYSTEM

GIVE A NEW DIMENSION TO PERSONAL COMPUTING. . .The amazing **Voice Master Key System** adds recognition to just about any program or application. Voice command up to 256 keyboard macros from within CAD, desktop publishing, word processing, spread sheet, or game programs. Fully TSR and occupies less than 64K. Instant response time and high recognition accuracy. Voice recognition tool-box utilities are included. **A genuine productivity enhancer!**

SPEECH RECORDING SOFTWARE. . .Digitally record your own speech, sound, or music to put into your own software programs. Software provides sampling rate variations, graphics-based editing, and data compression utilities. Create software sound files you can add to macros for voice recognition verification response. **a complete, superior speech and sound development tool.**



SOFTWARE CONVERSION CODES. . .The **Voice Master Key System** operates a growing list of third party talking software titles using synthesized phonetics (text - to - speech) or digitized PCM, ADPCM, and CVSDM encoded sound files. **Voice Master Key system does it all!**

EVERYTHING INCLUDED. . .**Voice Master Key System** consists of a plug-in card, durable lightweight microphone headset, software, and manual. Card fits any available slot. External ports consist of mic inputs and volume controlled output sockets. **High quality throughout, easy and fun to use.**

ONLY $149.95 COMPLETE

ONLY $89.95 FOR TANDY 1000 SL/TL MODELS SOFTWARE ONLY—REQUIRES TANDY BRAND ELECTRET MICROPHONE.

ORDER HOTLINE: (503) 342-1271 Monday-Friday, 8AM to 5 PM Pacific Time

Visa/MasterCard, company checks, money orders, CODs (with prior approval) accepted. Personal checks subject to 3 week shipping delay. Specify computer type and disk format (3½" or 5¼") when ordering. Add $5 shipping charge for delivery in USA and Canada. Foreign inquiries contact Covox for C & F quotes. 30 DAY MONEY BACK GUARANTEE IF NOT COMPLETELY SATISFIED. ONE YEAR WARRANTY ON HARDWARE.

CALL OR WRITE FOR **FREE** PRODUCT CATALOG

COVOX INC.
675 Conger Street, Eugene, OR 97402
TEL: 503-342-1271 • FAX: 503-342-1283

Number **277** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

4

Abrash code, continued

```
    statement to no operation */
    *ScreenPtr |= 0xFF;
}

/* Draws a circle of radius Radius in color Color centered at
 * screen coordinate (X,Y) */
void DrawCircle(int X, int Y, int Radius, int Color) {
    int MajorAxis, MinorAxis;
    unsigned long RadiusSqMinusMajorAxisSq;
    unsigned long MinorAxisSquaredThreshold;

    /* Set drawing color via set/reset */
    outpw(GC_INDEX, (0x0F << 8) | SET_RESET_ENABLE_INDEX);
                                    /* enable set/reset for all planes */
    outpw(GC_INDEX, (Color << 8) | SET_RESET_INDEX);
                                    /* set set/reset (drawing) color */
    outp(GC_INDEX, BIT_MASK_INDEX);
                                    /* leave the GC Index reg pointing to
                                       the Bit Mask reg */

    /* Set up to draw the circle by setting the initial point to one
       end of the 1/8th of a circle arc we'll draw */
    MajorAxis = 0;
    MinorAxis = Radius;
    /* Set initial Radius**2 - MajorAxis**2 (MajorAxis is initially 0) */
    RadiusSqMinusMajorAxisSq = Radius * Radius;
    /* Set threshold for minor axis movement at (MinorAxis - 0.5)**2 */
    MinorAxisSquaredThreshold = MinorAxis * MinorAxis - MinorAxis;

    /* Draw all points along an arc of 1/8th of the circle, drawing
       all 8 symmetries at the same time */
    do {
        /* Draw all 8 symmetries of current point */
        DrawDot(X+MajorAxis, Y-MinorAxis);
        DrawDot(X-MajorAxis, Y-MinorAxis);
        DrawDot(X+MajorAxis, Y+MinorAxis);
        DrawDot(X-MajorAxis, Y+MinorAxis);
        DrawDot(X+MinorAxis, Y-MajorAxis);
        DrawDot(X-MinorAxis, Y-MajorAxis);
        DrawDot(X+MinorAxis, Y+MajorAxis);
        DrawDot(X-MinorAxis, Y+MajorAxis);
        /* Advance (Radius**2 - MajorAxis**2); if it equals or passes
           the MinorAxis**2 threshold, advance one pixel along the minor
           axis and set the next MinorAxis**2 threshold */
        if ( (RadiusSqMinusMajorAxisSq -=
            MajorAxis + MajorAxis + 1) <= MinorAxisSquaredThreshold ) {
            MinorAxis--;
            MinorAxisSquaredThreshold -= MinorAxis + MinorAxis;
        }
        MajorAxis++;            /* advance one pixel along the major axis */
    } while ( MajorAxis <= MinorAxis );

    /* Reset the Bit Mask register to normal */
    outp(GC_INDEX + 1, 0xFF);

    /* Turn off set/reset enable */
    outpw(GC_INDEX, (0x00 << 8) | SET_RESET_ENABLE_INDEX);
}
```


MISOSYS

Programmer's Journal downloads are now available from MISOSYS' forum on Compuserve: GO PCS49

Are you still fussing with floppies for BACKUP? Our QIC40 JUMBO tape drive from MISOSYS is your solution!

The Colorado Memory Systems' JUMBO tape drive is one drive that fits all computers. It comes ready for internal use in AT's, XT's, and PC's, and connects to your floppy disk controller. Our adaptor board can be used to connect JUMBO when all FDC connectors are in use. Kits are available for XT/AT or PS/2 machines which convert Jumbo to external use.

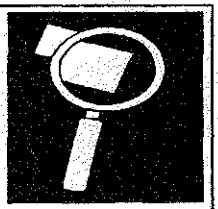
- In about 5.5 minutes, JUMBO backs up 10MB's file-by-file - the fastest in the industry! 40MB's gets backed up in about 18 to 20 minutes. Uses industry-standard DC2000 tape cartridges.
- JUMBO plugs into your floppy disk controller to save cost, power, and a slot. Needs 5-1/4" (or 3.5" with faceplate) mounting slot.
- Our tape adapter board mounts in your host computer to provide an additional tape port — especially useful in 286 and 386 computers. When used with our external JUMBO, it lets you share your drive between computers. Note: external adaptor includes "Tape Adaptor"
- JUMBO has custom chips, high speed brushless motors, automatic circuits, no pots to vibrate out of calibration, and fewest parts to assure long life and technical leadership for years to come.

DJ10 Jumbo	$295 ($5S&H)
Tape Adaptor	$75 ($3S&H)
External Adaptor	$110 ($5S&H)
DC2000 tape	$22.50

M/C & VISA. S&H are U.S. only.

MISOSYS, Inc.
P.O. Box 239
Sterling, VA 22170-0239
800-MISOSYS
or 703-450-4181

Now get the clearest view of your disk drive and memory contents with our DED86™



Powerful features in Version 2!

When you need to travel through your disk drive, why settle for a tool that isolates you from the road? DED86 gives you the direct controls you need to explore your disk. It's a full-screen sector-oriented disk/file editor and a page-oriented memory editor. When you want to "unerase" erased files, DED86's flexible KEEP facility does the job without you fussing over FATs.

Check out this list of features

- ✓ Look by cylinder/head/sector, sector or cluster; even reserved/hidden sectors
- ✓ Scan free clusters to search for erased data ✓ Jump about subdirectories
- ✓ Keep sectors & clusters for writing to a file; Great for partial file recovery
- ✓ Edit bytes in hexadecimal or ASCII, zap in 0s; Undo last edit!
- ✓ Search your disk drive, a file, or memory for ASCII or hexadecimal strings
- ✓ Touch a directory file entry with your date and/or time
- ✓ Obtain complete disk statistics in one screen; includes usage data
- ✓ Alter file attributes: archive, system, hidden, read
- ✓ Save/Restore sectors to/from auxiliary buffers; for moving data around
- ✓ DOS subshell available while using DED86 lets you shell to other programs
- ✓ Handles 5.25" & 3.5" drives & RAM disks; even large partition drives

DED86 reduced by more than 50%; Now only $29.95 + $2S&H

Fujitsu Floppy Disk Drives:

5.25" 360K 1/2 height	$75+$4S&H
3.5" 720K in 5.25" frame	$85+$3S&H
3.5" 1.44M in 5.25" frame	$99+$3S&H

Number **121** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

A Net Gain: NFS, Network File System

by

Frank D. Greco

Nowadays, most PCs and workstations are connected to some sort of network. The era of standalone PC computing (the "Me" generation) has given way to network computing (the "Us" generation) and thankfully, is pulling us away from proprietary IBM mainframe computing (the "Them" generation). Our workstations are all interconnected with different types of machines: the company IBM MVS mainframes, the department DEC VAX/VMS minis, the increasing number of SunOS-based SPARC computers, and the hordes of MS-DOS PC-class machines. And they are all Etherneted, token-ringed, and perhaps FDDIed together in one morass of wiring disorder. Wouldn't it nice if, from the PC in front of you, you could access files from each of these machines as easily as you access files from your local disk drive? The Network File System (NFS) makes this all possible.

Without using NFS, you could transfer a copy of the remote files to your local machine using some sort of communications program and perhaps perform some magic incantation to ensure the correct byte ordering. You could also log in to the remote machine to get at the files located there. It's either a duplication of data or just plain clumsy.

Whaddya Spec?

The Network File System provides largely transparent access to files and file systems across a heterogeneous (that is, mixed-vendor and mixed-architecture) network. While it is a public domain specification defined by Sun Microsystems, NFS has been implemented or licensed by more than 150 vendors.

NFS was not implemented as part of a single distributed operating system, à la the Locus (MIT) or Plan-9 (Bell Labs) operating systems. Rather, it was created as an interface to a variety of operating systems. The purpose of this open design was to eliminate the reliance on only one operating system and interconnect to different vendors' operating systems. Once an NFS directory (subtree) is "mounted" onto the current file system from across a network, it allows users and programmers to treat these remote files as if they were local.

Clear Transparency

NFS is a network-distributed file system that is transparent to application programs in several ways. It's easy to see its transparency.

First, NFS allows an operating system to access completely different types of file systems on other operating systems; so, NFS is operating-system and file-system independent. You can mount Sun SunOS file systems onto a PC using an MS-DOS file system. You can mount DEC VMS file systems onto a Sun using SunOS. Given the appropriate security permissions, as an NFS client (a requestor of NFS services) you can mount any NFS server (a deliverer of NFS services), regardless of operating system. I've even worked on NFS file systems for optical disk jukeboxes!

Next, since NFS does not differentiate between remote files and local files, the data itself can reside anywhere. An NFS file system can be on the same hard disk or on a disk across the network. A colleague of mine told me of an interstate NFS mount over the airwaves using packet radio, which I thought was pretty neat. It wasn't very fast compared with an Ethernet local area network, but, hey, it worked (at 56Kb). Now that AT&T is starting to commercialize its high-speed, satellite-based, wide area network, it will soon be possible to NFS-mount a file system in London onto a machine in New York. At the last SIGGRAPH, AT&T and Sun were demonstrating this technology to offer distributed computing over a large geographic area (they were showing real-time, distributed visualization via satellite from a Cray to a SPARC).

Makes No Difference to Me

The transparency of NFS also extends to machine architectures. So, on a 386-based PC, one can mount an NFS file system from a RISC-based SPARC machine, an NFS file system from a 68Kb-based file server, and an IBM MVS mainframe and

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

exchange files—all with no regard to byte ordering! And NFS can be implemented over any type of network or internet protocols, not just Ethernet.

Better than Fine Corinthian Leather

I recall a demo that I conducted recently for an important client. I mounted a DEC VMS file system, a Prime PrimOS file system, an IBM MVS file system, an AT&T System V file system, a Berkeley 4.3 file system, and several SunOS file systems, all from a color SPARCstation using NFS. If that wasn't impressive enough, I threw in a live video feed in a window for good measure (no, I wasn't showing Oprah, Phil, or Geraldo) along with a couple of concurrent 3270 sessions in some other windows; needless to say, the mainframers in the crowd were slack-jawed. The point of this was not that "Frank gives good demo"; it was that interoperability among vendors and operating systems is available now.

The NFS protocol is considered to be stateless, meaning that neither the file client nor the file server maintains any sort of data that indicates state. So, your favorite system administrator does not have to stitch things back together if the file server crashes. The NFS client workstation merely loops, continuing its attempt to re-establish its connection with the NFS server. Because of the lack of state (and any subsequent buffering), any NFS write to a disk over the network is synchronous; disk reads, on the other hand, merely read from the standard UNIX disk cache (normally called the buffer cache). That means that a write has to complete physically before returning across the network to the originating client application program. The only disadvantage of this scheme is that while NFS reads are fairly fast, NFS writes can be slow compared with a local disk. If you're sending very large, multimegabyte files, this could be costly.

However, there are several vendors out there—Legato comes to mind—that are throwing hardware at this problem. Legato has a battery-backed board (called "PrestoServe") that logically completes the NFS disk write but physically stores the data in a RAM cache on the Prestoserve board. Then, at the right moment, the data is written out to the disk in physically optimal locations for subsequent speedy access.

The era of stand-alone PC computing (the "Me" generation) has given way to network computing (the "Us" generation). . . .

We're DOS in the Wind

For those PC programmers out there who are not familiar with logically manipulating physical data, let's consider the MS-DOS utilities **assign** and **join**. These two commands are rarely used by MS-DOS programmers and users, but they are useful—if not vital at times. When I recently installed Ventura on my Wyse 3216, I *had* to use **assign**. I had ordered Xerox Ventura Publisher on 3.5-inch diskettes so that I

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

could use my newly installed Toshiba 3.5-inch drive, which I set up as my B drive.

I inserted the Ventura installation diskettes and ran the Install program. However, the Ventura people didn't have the foresight to consider this type of installation. Their installation procedures assumed an A drive. So, the Ventura Install program kept on looking for a disk on my 5.25-inch A drive, which, of course, it did not find. And you know how aggravating it is when a program attempts to access a drive and there's no disk there: You get that stupid "Abort, Retry or Ignore" error message. And you *must* put a floppy in that drive to get out of this insane predicament. (Great design, isn't it?)

Well, the solution to my problem was just to say:

```
C:\> assign b: a:
```

So, from now on, any operations involving the A drive now use the B drive instead. Once I did that, the installation of Ventura went smoothly. (A much better way would be to have Ventura make intelligent decisions as to what drive to install from or, if it couldn't, to ask the user for additional information, for Pete's sake.)

The other MS-DOS command, **join**, might give you more of a flavor of what NFS offers. Try the little MS-DOS experiment provided in the box to see the names of your files on the disk in your A drive.

Join logically "mounts" a drive onto a directory (the so-called "mount point"). So, whenever you (logically) access that directory, you are (physically) accessing that disk drive. Unfortunately there are limitations.

Join cannot extend over a network, it cannot connect two directories (you can mount a drive only on a directory), and it does not work with many other MS-DOS utilities. Nice try, but no cigar. (Reminds me of my CIS senior undergrads with their semester projects.)

How It Woiks

The NFS protocol sits on top of two other protocols, RPC and XDR, which can be used separately if you wish. These are all members of Sun's freely licensable Open Network Computing (ONC) environment.

RPC is Sun's Remote Procedure Call protocol specification. It is a mechanism by which procedure calls are transparently executed across a network. A process on one machine can have a process on another machine execute a procedure as if the original process executed it locally. RPCs are the heart of NFS.

```
C:\> md \adrive         # Create a temporary directory
C:\> join a: \adrive    # "Mount" the A drive on this directory
C:\> dir \adrive        # Perform any operation on this directory
```


Multitasking

Do the multitasking systems you are considering using seem a bit like over kill or need an awful lot of memory? Maybe all you really need is a system which can run more than one DOS program concurrently. If this sounds like your situation, consider MultiDos Plus. For $24.95 or $99.95 (if you want our source code) you can run up to 32 tasks and still use MS-DOS or PC-DOS. For more information please call or write for a free brochure.

Nanosoft
800-678-2141
508-651-0091
FAX 508-655-8860
13 Westfield Road
Natick MA 01760

Number 109 on the Reader Service Card

A professional workstation environment for OS/2™

Hamilton C shell™

The superior alternative to the standard OS/2 command processor. Faithfully recreates and updates the entire C shell language as described in the Berkeley 4.3 *UNIX® Programmer's Manual*. Created explicitly for OS/2 using modern incremental compiler technology. Not one line ported from or created on anything but OS/2. Very high performance. Extensive support for multi-threading.

Features: Very long command lines: 64KB = 10x the UNIX limit • Fully recursive filename wildcarding: complex patterns, multiple directory-level matches • Fully nestable `if`, `foreach` and other control structures • Variables, arrays and a wide range of expression operators and built-in functions • History for recalling and editing past commands • Faster interrupts, I/O redirection and piping • Command substitution: use the output of one command as command-line arguments to another • Aliases and shell procedures • Filename completion • Arrow keys • Hashing to speed searches of the `PATH` directories • Background threads and processes, `ps` and `kill` • Numerous professional-quality utilities.

Requires OS/2 1.1 or later. All executables will run properly inside a Presentation Manager window. Not copy-protected.

$350.00. Unconditional satisfaction guarantee.

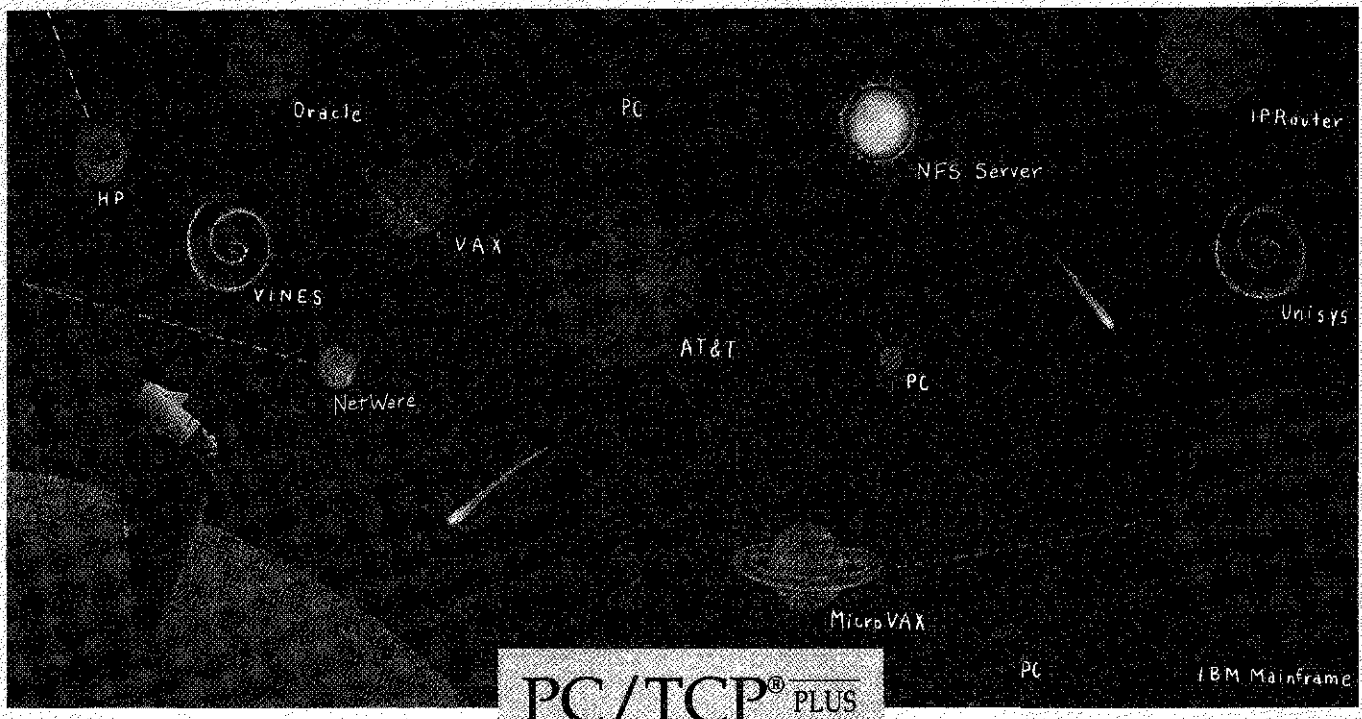
Phone 508-358-5715. MasterCard and Visa accepted.

Hamilton Laboratories
13 Old Farm Road, Wayland, MA 01778–3117

Number 334 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

There's only one way to get around out here.



PC/TCP® PLUS

The most comprehensive TCP/IP implementation in the industry.

SUPPORTS NFS!

No matter who or what your PC needs to communicate with, there's nothing like PC/TCP Plus. It's not only the fastest, but also the most comprehensive TCP/IP implementation in the industry. And PC/TCP Plus has NFS™ access built right in.

Providing connectivity to other PCs, minicomputers and mainframes, PC/TCP Plus communicates over Ethernet, Token Ring and StarLAN networks with operating systems ranging from UNIX and VMS to VM and MVS. And you can forget hardware incompatibility; PC/TCP Plus supports the largest selection of LAN interfaces in the business.*

From around the world to around your office, now your PC can transfer files, send and receive electronic mail, emulate VT100, VT220 and IBM 3278 terminals, access NFS servers, and much, much more. VINES™ and NetWare™ users will appreciate PC/TCP's full compatibility, as will customers of Oracle's distributed PC database products. And with our Development Kit, including a Berkeley Sockets library, you have all you need to produce custom applications.

So, the next time you want to get from here to there, get PC/TCP Plus, the TCP/IP implementation that gives you a whole new perspective on network communication. Call us at (617) 246-0900 for more information.

PC/TCP is a registered trademark of FTP Software, Inc.
VINES is a trademark of Banyan Systems, Inc.
NetWare is a trademark of Novell, Inc.
NFS is a trademark of Sun Microsystems, Inc.

FTP Software, Inc.
26 Princess St.
Wakefield, MA 01880-3004
Phone: (617) 246-0900
Fax: (617) 246-0901

ftp Software, Inc.®

*PC/TCP Plus supports interfaces from Acer, Allied Telesis, Apricot, AT&T, BICC, DEC, D-Link, DSC, Excelan, Gateway Communications, IBM, IMC Networks, Intel, Interlan, Longshine, MCAssociates, National Semiconductor, Novell, Proteon, Schneider & Koch, Scope, 10Net, 3Com, Tiara, Torus, TRW, Ungermann-Bass, Univation, Western Digital and YCS, in addition to the NDIS and Packet Driver specifications.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Recently, an organization of vendors (including Sun, Novell, Netwise, 3Com, Lotus, Sybase, Unify, Informix, Banyan, Relational Technologies, Ashton-Tate, AST Research, Interactive Systems, Microrim, Oracle, and Wang) endorsed Sun's RPC mechanism as a standard method of intermachine communication, also known as "distributed computing." Novell and Netwise are teaming up with Sun to develop even-higher-level mechanisms to perform distributed computing over a series of mixed-vendor computer networks.

An eXcellent Adventure

The other part of the underlying NFS machinery is XDR, an acronym for eXternal Data Representation. XDR offers a standard means of exchanging data among heterogeneous machines, usually across a network. It offers a standard architecture-independent format for byte/bit ordering, floating-point representation, and word length. Data is converted to XDR format from whatever your local machine's architecture, sent over the network in XDR format, and then converted to the target machine's format on the target machine by the target machine. XDR is how NFS handles the different machine-architecture dependencies; so, when you use NFS, you indirectly use XDR.

Now, you don't necessarily have to use XDR remotely over the network; you can use it locally, too. Suppose you have several different types of workstations sharing a local disk—some SPARC machines, some Sun-3 68030 machines, and some 386-based PCs. If you want to maintain an architecture-independent file format for all three different machine architectures, you can use XDR. Without XDR, files created by one machine might not be easily readable by another machine, because of obvious byte-ordering dependencies. Or, if the machines are all connected via a network and NFS is available, you can merely NFS-mount the directories and let NFS handle the XDR translation at a higher level.

Sockets, Pipes, and Insulation

Along with Sun, Netwise has developed a series of high-level RPCs that will allow programmers to develop distributed applications over a series of different computer networks. Right now, the RPC mechanisms are built over the Berkeley-based socket paradigm; in the future, the RPC interfaces will be built over the Transport-Layer Interface (TLI), which will be available in both the upcoming UNIX System V.4 and Berkeley 4.4 operating systems. Using the TLI will insulate the application from the underlying transport mechanism. So, now programmers could use this enhanced RPC interface to write portable distributed applications without knowing if the underlying communication is using BSD sockets, MS-DOS NetBIOS, or OS/2 named pipes. Sun has placed the specification for this enhanced RPC in the public domain. The source code to the original RPC interface, which is used by Sun's ONC (Open Network Computing) licensees, is available free of charge over the ever-useful and free-access Usenet network.

Local Boy Connects in Big City

For the PC crowd, there are several implementations of NFS for PCs. One of them is PC-NFS, a product from Sun. PC-NFS is a client-based implementation of NFS for PC-class machines using MS-DOS. Besides Sun, there are at least two other vendors selling NFS for the PC. If you get the PC-NFS developer's package from Sun, you also get the rsh, rlogin, telnet, and ftp high-level user utilities and programmatic access to the lower-level RPC calls.

Using the rsh (remote shell) utility, you can run a program on another machine on the network but have its output appear on your PC (actually standard output). I've used this feature plenty of times to invoke compute-bound tasks on a SPARC machine but have its output come back to the PC. Of course, access to the lower levels of NFS is provided via the RPC and XDR interfaces, which come along in a series of library calls if you want to use them directly.

Older versions of PC-NFS could not be used with popular programs

People are talking about us.

F77L-EM/32
Port 4GB mainframe programs to 80386s with this 32-bit DOS compiler. Winner of *PC Magazine's* 1988 Technical Excellence Award. $895*

F77L
The compiler of choice among reviewers and professionals. New Version 4.0 includes an Editor, Profiler, Linker, Make Utility, Weitek and 386 Real-Mode Support, Graphics. $595

F77L-EM/16
Address up to 15MB on 80286s with this award winning extended-memory compiler. $695*

Lahey Personal FORTRAN 77
Full ANSI77, Microsoft C and Borland C interfaces, Debugger, at an unbeatable price. $95

*Requires DOS Extender ($195)

Contact us to discuss our products and your needs. **(800) 548-4778**
Lahey Computer Systems, Inc. P.O. Box 6091, Incline Village, NV 89450
Tel: (702) 831-2500 FAX: (702) 831-8123 Tlx: 9102401256
FORTRAN IS OUR FORTE

Number 44 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

such as Microsoft Windows because of software-interrupt collisions, but now you can use NFS with the low-end PC windowing system. I've used this capability to design PC-based imaging systems using the simplistic, nonpreemptive multitasking capabilities of MS Windows and the enormous disk capacity of a-couple-of-gigabyte NFS servers over the Ethernet.

Unfortunately, while you can use a PC as an NFS client to get files from your Sun or DEC VAX, you cannot use your PC as an effective NFS file server. Since MS-DOS is a single-tasking "operating system" (I still cringe when I say that . . . it's not really an operating system), the PC would have to handle network requests for doling out files in a serialized fashion. Since there can be many, many requests for file service over a network, the inefficiencies and ineffectiveness of this scenario under DOS are clear. Until an OS/2 implementation of NFS is completed (Sheesh, who knows when *that* will be!), don't look for PC-based NFS servers soon.

In any event, NFS-based file servers are going to be around for a long time. The idea is just too good.

La Machine

Now that NFS is an accepted technology, think of the commercial-product potential of this idea (venture capitalists out there, are you listening?): a little PC-type of computer controlling a relatively large, 300–500 megabyte ESDI or SCSI drive (or even one of those new Maxstor 1.2-gigabyte, 5.25-inch drives), using a custom-built, streamlined operating system that just handles the NFS protocol. An NFS-Machine! You wouldn't need Unix at all. The operating system would be optimized to use lightweight processes, since there would be no need for a generalized-process mechanism. There would also be no need for process-memory protection, since it isn't a general-purpose machine used for non-NFS purposes. So, this operating system could be high speed indeed.

You could probably whip up a mini-operating system like this in no time. And since the throughput of this system would really depend on the disk controller and not on the main processor, even a 286 would probably be able to handle the job. Neat idea, isn't it? A low-cost NFS-server. And just the right size for workgroup computing, too. Ah, isn't science wonderful, kids? PJ

Frank D. Greco, president of Crossroads Technologies, Inc., performs industry analysis, systems consulting and training, and technological forecasting for Fortune 500 companies. Readers can contact Frank at Crossroads Technologies, Inc., P.O. Box 530, Fanwood, NJ 07023, 201-756-3075.

References

Sandberg, Russell, "Design and Implementation of the Sun Network File System," Usenix Proceedings, Portland, Summer 1985.

Trademarks

UNIX is a trademark of AT&T Bell Laboratories. NFS and SPARC are trademarks of Sun Microsystems.

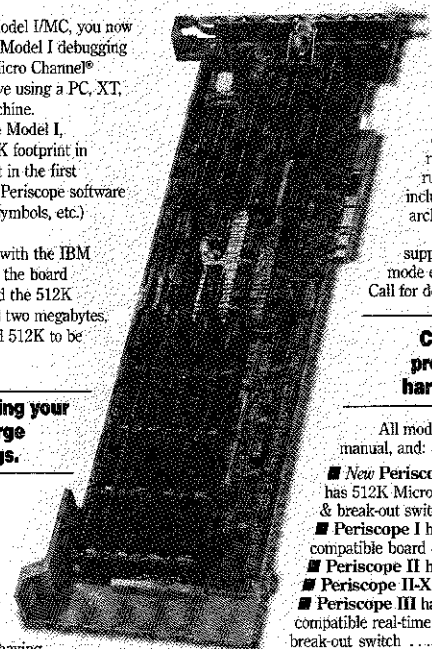
[redacted]

Now There's a Periscope Board for Your IBM PS/2®

With the new Periscope® Model I/MC, you now have the same robust Periscope Model I debugging capabilities using a PS/2 with Micro Channel® architecture that you already have using a PC, XT, AT, or AT-compatible 80386 machine.

Just like the current Periscope Model I, Periscope Model I/MC has a 32K footprint in system memory, above 640K but in the first megabyte. The board stores the Periscope software and all debugging information (symbols, etc.) in its write-protected RAM.

Designed for use in machines with the IBM Micro Channel bus architecture, the board allows you to add chips to extend the 512K of write-protected RAM to a full two megabytes, if need be. (Most developers find 512K to be quite enough, however.)



Real-time hardware-assisted debugging of programs running on PS/2s is now possible! The remote feature of the new Version 4.3 Periscope software enables Periscope IV to support real-time debugging of programs running on DOS-based machines, including those with Micro Channel architecture. The open architecture remote debugging feature will support OS/2® and other protected-mode environments in the near future. Call for details.

Don't worry about trashing your debugger, debugging large programs, or erratic bugs.

With this new board in your IBM PS/2 or compatible, Periscope uses zero memory in the lower 640K. So you don't have to worry about things like a runaway program trashing your debugger, or not being able to debug a very large program, or having bugs appear or disappear when you load your debugger.

Use the break-out switch, which plugs into the board, to break in to your system safely any time. It keeps you from having to power down and back up when your system hangs. You can just press the little red "panic" button to find out exactly what is going on.

Periscope Model I for PCs, XTs, ATs, and AT-compatible 80386s. The manual, disk, and quick-reference card shown come with all models of Periscope.

Choose from a full line of professional software and hardware-assisted models.

All models include Version 4.3 software, manual, and:

- *New* **Periscope I/MC** (MC Board for short) has 512K Micro Channel-compatible board & break-out switch ..... $745.
- **Periscope I** has 512K PC- and AT-compatible board & break-out switch ..... $595.
- **Periscope II** has break-out switch ..... $175.
- **Periscope II-X** has no hardware ..... $145.
- **Periscope III** has PC- and AT-compatible real-time board (to 10MHz) & break-out switch ..... $1395.
- **Periscope IV** has 80286 and 80386 AT-compatible real-time hardware (to 25MHz) & breakout switch ..... $2195-$2995.
- **PLUS** board is Model I board (no software), optional with Models III & IV ..... $500.

Call Toll-Free Today For More Information 800-722-7006

MAJOR CREDIT CARDS AND QUALIFIED COMPANY PURCHASE ORDERS ACCEPTED

IBM, PS/2, OS/2, and Micro Channel are registered trademarks of the IBM Corporation.

PC TECHNOLOGY 1990 FINALIST

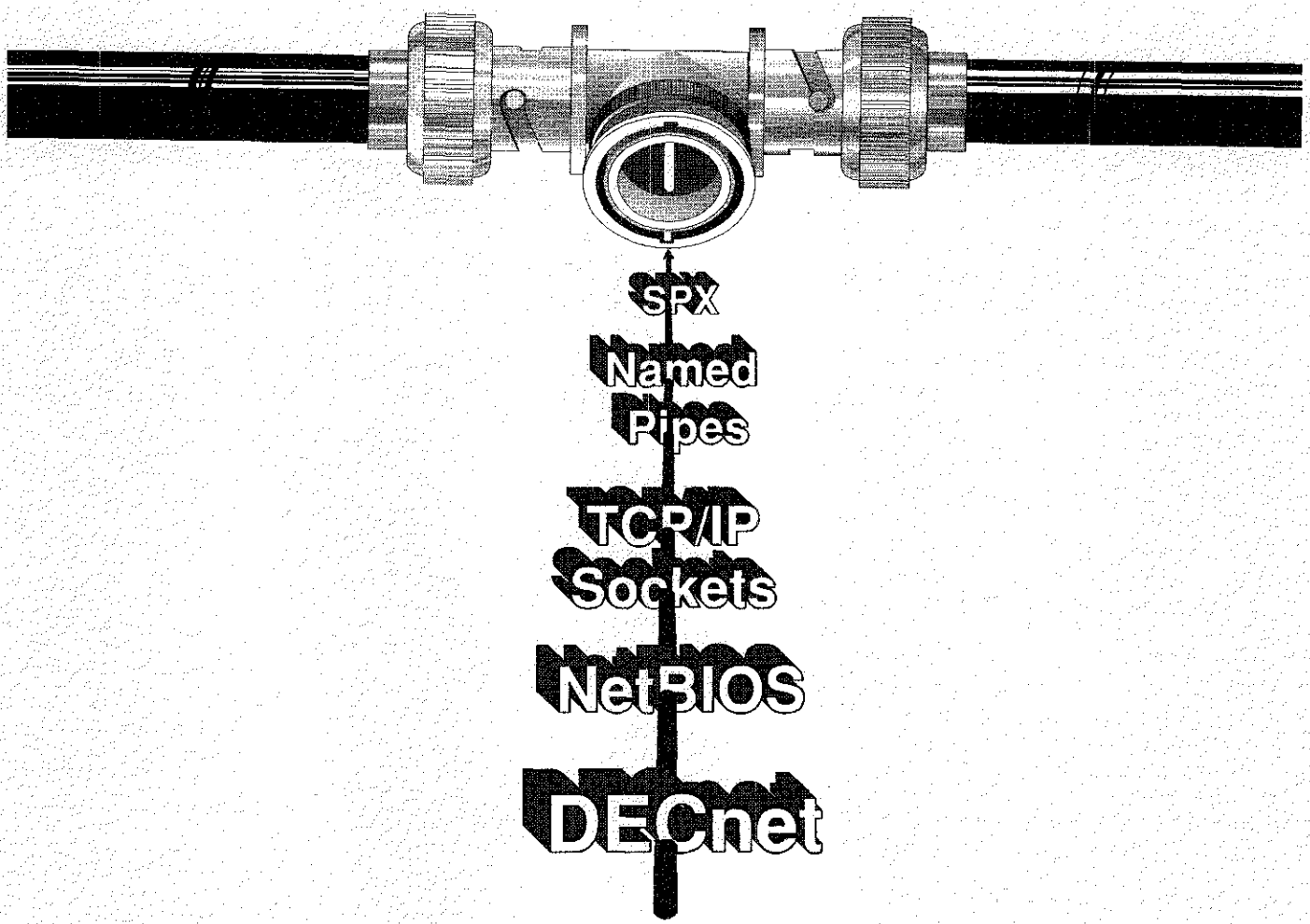
1197 PEACHTREE ST. • PLAZA LEVEL • ATLANTA, GA 30361
404/875-8080 FAX 404-872-1973

The Periscope Company, Inc.

Number 6 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Are networks creating obstacles for your applications?



Writing network communications software for your applications is difficult enough. Writing it for multiple networks or to connect applications running on different operating systems and hardware platforms is just too much. Too much time. Too much money. And too much energy searching for the networking software expertise you need to overcome your multi-vendor computing obstacles.

At Netwise, we believe your time and money are best spent at the applications level, building solutions for today's network computing environments — not struggling with writing low-level networking code.

The Netwise® RPC TOOL™ lets you breeze past the obstacles by providing access to a common software platform for the development of distributed applications. You write the application, then specify the network. The RPC TOOL automatically generates distributed processing software for you. All of it. Every time. And does it TODAY — for NetBIOS, NetWare, LAN Manager, DECnet, and TCP/IP networks. No waiting on promises. No vaporware. No excuses.

Banyan, Novell, Sun, 3Com and other major software and hardware suppliers have endorsed this technology as a standard for building distributed applications. Now it's your turn.

Call us for a FREE educational diskette. Or order a PC version of the RPC TOOL for a FREE thirty-day trial. Save your struggling for obstacles worthy of the effort.

Use the Netwise RPC TOOL once, and never write network communications code again.

Netwise is a registered trademark and RPC TOOL is a trademark of Netwise, Inc. Banyan is a registered trademark of Banyan Systems Incorporated. DECnet is a trademark of Digital Equipment Corporation. Novell is a registered trademark and NetWare is a trademark of Novell, Inc. Sun is a registered trademark of Sun Microsystems, Inc. 3Com is a registered trademark of 3Com Corporation.

NETWISE®

Call us. 800/733-7722
2477 55th Street,
Boulder, Colorado 80301
USA

Call me, I'm interested: Circle 378 **Send Literature: Circle 278**

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Exploiting NetBIOS

by
W. David Schwaderer

Adapted from C Programmer's Guide to NetBIOS *(Howard W. Sams & Company, 1988)*

The Network Basic Input/Output System (NetBIOS) is an application programming interface for data exchange between data sources and data sinks. Loosely speaking, NetBIOS is a programming gateway to sets of services that allow computer applications and devices to communicate. Application programs must generally invoke these various NetBIOS services using specific command sequences. Hence, NetBIOS has explicit, though minimal, protocols associated with some of its services.

Typically, data exchange occurs between NetBIOS applications residing within separate machines connected by a Local Area Network (LAN). However, two applications within the same machine can also use NetBIOS for data communication without a LAN. Thus, although all IBM NetBIOS implementations require a LAN adapter, NetBIOS use is not restricted to LAN environments.

If you are familiar with data communication theory, you might recall the International Standards Organization (ISO) Open Systems Interconnection (OSI) Reference Model. The OSI model divides the various activities required for orderly data communication between two applications residing in distinct machines into seven discrete processes, or "layers." During application-to-application communication, each layer within a given machine directly coordinates message-passing activities with the adjacent layers immediately above and below it.

NetBIOS is situated high within the reference model hierarchy. Hence, applications that program to the NetBIOS interface are largely isolated and essentially insulated from the precise way the lower layers interact with their peer and adjacent layers.

NetBIOS first appeared in August 1984 with the IBM PC Network Adapter Card designed by Sytek Inc. Located on this IBM PC Network adapter (LANA) is an extended BIOS ROM referred to as the LANA's Network Adapter BIOS. This ROM module occupied 8Kb of memory, starting at memory segment CC00h, and contains initialization routines, diagnostics, coprocessor and PC memory interface routines, and part of the first NetBIOS implementation. The remainder of this NetBIOS implementation was located on a second adapter ROM referred to as the adapter's Protocol ROM. Nowadays, most network software uses NETBIOS.COM, a complete NetBIOS software replacement for the PC Network LANA NetBIOS.

Application Services

NetBIOS provides four categories of application services:

- Name support
- Datagram support
- Session support
- General commands

Name Support

An individual NetBIOS LAN adapter is distinguished from other adapters on its respective network by one or more network names, which allow LAN applications to direct their messages to specific adapters and indicate that their adapter originated the message.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NetBIOS Commands

Name Support
- Add Name
- Add Group Name
- Delete Name

Datagram Support
- Receive Datagram
- Receive Broadcast Datagram
- Send Datagram
- Send Broadcast Datagram

Session Support
- Call
- Listen
- Send
- Send No-Ack
- Chain Send
- Chain Send No-Ack
- Receive
- Receive Any
- Hang Up
- Session Status

General Commands
- Reset
- Cancel
- Adapter Status
- Unlink

Each network name consists of 16 characters. Within a network name, each of the 16 characters is significant, and uppercase is different from lowercase. The names you can create cannot have a value of binary zero or an asterisk (*) as the first character. IBM reserves the values of 00h to 1Fh for the sixteenth character and uses some of the reserved character values with the IBM PC LAN Program. This is why you can have only a 15-character IBM PC LAN Program machine name. Finally, IBM reserves the use of "IBM" as the first three characters of any name.

The number of names an adapter can use (or is using) will vary, as can the number of adapters using a given name. But before an adapter can use any name, it must acquire the rights to register and use the name on the LAN.

NetBIOS initiates name-registration activities in response to either of the two types of NetBIOS add-name commands: Add Name and Add Group Name. An adapter registers a network name by first broadcasting a network petition (a name-claim packet) to use the name. The type of packet, Name_Claim or Add_Group_Name_Claim, indicates whether the adapter wants to register the name as a *unique name* or as a *group name*, respectively.

Finally, once a name is successfully registered, any registered name except the first can be deregistered by the issuing of a NetBIOS Delete Name command. NetBIOS adapter reset commands erase the NetBIOS name table (except the first name), as do a system reset (CTRL-ALT-DEL) and powering off the workstation.

If an adapter tries to register a name as a unique name, then no other adapter operating on the LAN can have the same registered name; otherwise, the registration attempt fails. If an adapter tries to register a name as a group name, then no other adapter can be using that name as a unique name; otherwise, the registration attempt again fails. This allows other adapters to register the name as a group name, though not as a unique name. Group names are useful for sending messages to collections of workstations, such as departments or teams.

If an attempted name registration fails, the failure is reported to the workstation application for subsequent analysis along with an appropriate error return code. In the absence of network complaints, the adapter's NetBIOS support places the name in a locally maintained internal table, known as the NetBIOS *name table*. It then reports the name-registration success to the LAN application along with a one-byte value.

The one-byte value is an unsigned number referred to as the name's NetBIOS *name number*. The name number is subsequently used in various NetBIOS commands associated with the name. NetBIOS assigns the value of the name number in an incremental, modulo 255, round-robin manner. The values 0 and 255 are never assigned, and the first entry is permanently assigned by the adapter based on its internal serial number. Thus, the numbers are assigned in the order 1, 2, 3, . . . 254, 2, 3, 4, . . . 254, etc.

Placing the name in the name table authorizes the adapter to subsequently scrutinize registration petitions of other adapters wishing to register names. And, once added, a name can be deleted from the table, potentially allowing some other name to use it as a unique name.

Note that the NetBIOS name table is a temporary table contained within RAM and is reconstructed after each system boot or adapter reset. Because each adapter has its own private name table, NetBIOS name resolution is highly autonomous across the LAN, requiring no central name administration. If a NetBIOS module is supporting more than one LAN adapter within a workstation, each adapter also has its own, independent NetBIOS name table.

Every IBM LAN adapter has a six-byte number associated with it, guaranteed to be unique for each adapter and contained in an adapter ROM. The number is referred to by a variety of names:

- Permanent node name
- Permanent node address
- Burned-in address (BIA)
- Universally administered address
- Unit identification number
- Physical address
- Local node name

For all IBM LAN adapters other than the PC Network LANA card, this address is in the range that is universally administered by the standards committees for local area networks and has the two high-order bits set to 0. Under the native ROM NetBIOS, PC Network LANA adapter cards always have two bytes of binary zeros as the last two bytes of their permanent node names. The values of the two high-order bits in the remaining four bytes vary by adapter. The permanent node name can be overridden at boot time with a six-byte locally administered address whose high-order bytes must have a value of X'4000'.

When any LAN adapter is initialized and active on a LAN, it has a unique six-byte number associated with it: the burned-in permanent node name. The number is also guaranteed to be unique on the LAN because it is registered in the NetBIOS name table as a unique name during adapter initialization (the 6 bytes are appended to 12 bytes of binary zeros to construct a unique 16-byte name). Because the registra-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tion happens during system initialization, the permanent node name is always the first entry in an adapter's NetBIOS name table. Since 0 is an invalid NetBIOS name number value, the permanent node name always has a NetBIOS name number value of 1.

The permanent node name serves as a LAN address that fingerprints all messages transmitted *by* an adapter and serves as an identification anchor point for all messages transmitted *to* an adapter. This unique 48-bit value constitutes an adapter's electronic-message sieve.

It would be convenient to personalize the LAN adapter address by using an associate's natural name (Melissa or Gregory, for example). Such pseudonyms are called *symbolic names* and are registered in the NetBIOS name table as either unique names or group names.

An adapter can receive messages that are addressed to it using only

- Its 48-bit unique address derived from its unique serial number
- An indiscriminate general broadcast address of X'FFFFFFFFFFFF'
- Bit-mapped functional addresses
- One value-mapped group address

The last two methods are beyond the scope of this article.

Once a symbolic name has been resolved into an appropriate 48-bit address, NetBIOS needs only the 48-bit address to conduct the communication. Symbolic names can be registered and deregistered.

Datagram Support

Once an adapter becomes active in a network, application programs within the workstation can use NetBIOS to communicate with other applications residing in the same or different workstations. The applications can communicate using either *datagrams* or *sessions*.

Datagrams are short messages whose size varies by NetBIOS implementation and that have no guarantee of delivery beyond a "best effort" by the adapter. Regardless of whether the messages arrive safely, no receipt indication is provided by NetBIOS. The intended recipient machine may

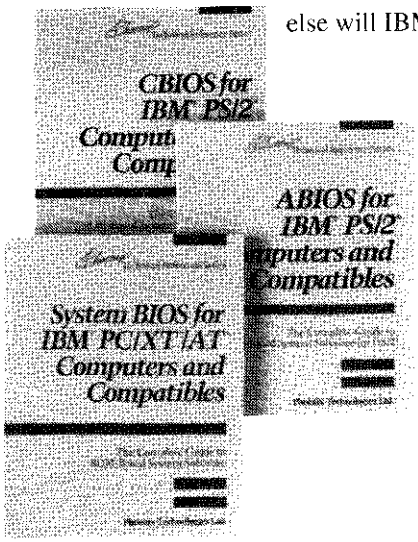
- Not exist
- Be powered off
- Not be expecting a datagram

In these instances, and in the case of network problems, the datagram may never be received by any workstation. In other words, datagram communication is "send and pray" communication unless the receiving application takes explicit action to transmit a receipt acknowledgement. The primary advantage of datagram communication is that it can consume less workstation resource than session communication.

There are two types of datagram communication: *broadcast datagrams* and *plain datagrams*. In both cases, the NetBIOS datagram transmission command references an existing local NetBIOS name number, perhaps the permanent node name's, that serves as the *datagram's origin name*.

Books from the BIOS Experts

The Phoenix Technical Reference Series gives programmers inside information on the ROM-based system software resident in IBM® PC, XT™, AT®, and PS/2® computers and compatibles. Because these books are written by the experts at Phoenix, they are comprehensive, definitive, and well written. No where else will IBM programmers find such accessible information on this vital topic.


CBIOS for IBM PS/2 Computers and Compatibles
ABIOS for IBM PS/2 Computers and Compatibles
System BIOS for IBM PC/XT/AT Computers and Compatibles

Books in the series include:

System BIOS for IBM PC/XT/AT Computers and Compatibles:
The Complete Guide to ROM-Based System Software
$26.95, 560 pages, 0-201-51806-6

CBIOS for IBM PS/2 Computers and Compatibles:
The Complete Guide to ROM-Based System Software for DOS
$26.95, 592 pages, 0-201-51804-X

ABIOS for IBM PS/2 Computers and Compatibles:
The Complete Guide to ROM-Based System Software for OS/2
$26.95, 688 pages, 0-201-51805-8

Available wherever computer books are sold.

Addison-Wesley Publishing Company
Route 128, Reading, MA 01867 (617) 944-3700

Number **331** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Broadcast datagrams are totally indiscriminate datagrams transmitted with a NetBIOS Send Broadcast Datagram command. Any adapter, including the transmitting adapter, can receive a broadcast datagram if it has previously issued a NetBIOS Receive Broadcast Datagram command. In general, broadcast datagram communication should be avoided, because two applications within the same workstation could easily receive broadcast datagrams intended for the other application.

Plain datagrams are discriminate datagrams transmitted with a NetBIOS Send Datagram command. Unlike with broadcast datagrams, applications specify a recipient NetBIOS name with the Send Datagram command. Any adapter, including the transmitting adapter, can receive a datagram if it has previously added the appropriate recipient name and issued a Receive Datagram command to NetBIOS referencing the number of the name specified in the command.

If an application specifies a name number of FFh in a Receive Datagram command, the application can receive a datagram for any name in the NetBIOS name table. This is referred to as a *receive-any* datagram. However, Receive Datagram commands for a specific name number have priority over Receive-Any Datagram commands. Finally, plain datagrams can be transmitted to adapters using the name as a unique name or to groups of adapters that share a group name.

Session Support

NetBIOS session support creates a reliable two-way data communication connection between two applications that can exist for extended periods. Such connections are sometimes referred to as *virtual circuits*.

The communicating applications may reside within the same workstation (local sessions) or within different workstations (remote sessions). Each application constitutes one *half* or *side* of the session.

The primary advantage of session communication over datagram communication is that message-receipt status is presented to the transmitting application for every message it transmits. However, session communication reliability comes with the slight overhead of creating and maintaining sessions and the packet-acknowledgement protocol between adapters.

Sessions are created when one application issues a NetBIOS Listen command referencing a name in its NetBIOS name table. The application may use an existing name in the table, such as the permanent node name, or add one of its own.

The Listen command also specifies the remote name that a petitioning application must use to qualify as a session partner and may use an asterisk (*) as the first character of the remote name. In this case, the remaining 15 characters are ignored and the local NetBIOS allows the second application to use any name to qualify as a session partner. (Since session security depends on matching both names, one might correctly suspect such promiscuous behavior has its hazards.)

A second application then issues a NetBIOS Call command, which references the name in its NetBIOS

Your operating system finally caught up with our editor.

Introducing BRIEF for OS/2.
It's the most versatile editor available, in a powerful new environment.

Now, you can have the best multi-level undo available, potent search and translate capabilities, and two powerful (but easy-to-use) macro languages. Plus all the inherent advantages of OS/2, like multitasking.

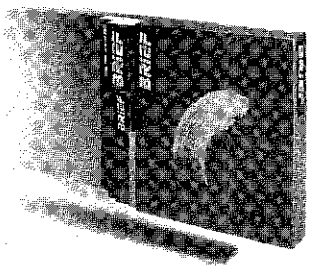
And BRIEF is still easy to use, and easy to learn. Commands are consistent and intuitive, and the documentation is clear and concise. Best of all, it's still affordable—just $195.

If you've made the investment in time and money to move up to OS/2, make sure you're working with an editor as advanced as your operating system.

Call today to order your copy of BRIEF for OS/2. And stop playing catch-up.

Special discount for DOS BRIEF users.

Call 1-800-343-7308.



BRIEF® for OS/2

UnderWare, Inc., 321 Columbus Ave., Boston, MA 02116 (617) 267-9743

Number **309** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

name table that the first application is expecting as a partner's name. The Call command also references the name the first application referenced in its own NetBIOS name table. The double name match fulfills the criteria of both applications to create a session, and the pending Listen and Call commands then complete. Note the sequence: first the Listen, then the Call. This sequence cannot be reversed successfully.

Each application then receives notification of session establishment and a one-byte unsigned value, referred to as the NetBIOS Local Session Number (LSN), that the adapter associates with the session. The LSN is analogous to a PC-DOS file handle.

NetBIOS assigns the LSN value in an incremental, modulo 255, round-robin manner. The values 0 and 255 are never assigned. Thus, the numbers are assigned in the order 1, 2, 3, . . . 254, 1, 2, 3, 4, . . . 254, etc.

Even if both sides of the session are local sessions, note that two numbers are assigned—one for each side. In this case, either application can use either LSN. In general, there is no restriction that the two LSNs have the same value, even if they are both local sessions.

After establishing a session, both sides can issue NetBIOS Send and Receive commands to transfer data. If a given name is used to create several sessions, an application can also issue a NetBIOS Receive-Any-for-a-Specified-Name (Receive-Any) command, which provides received data from any session associated with a specified name. More generally, the application can issue a NetBIOS Receive-Any-for-Any-Name (Receive-Any-Any) command, which provides received data from any existing session the adapter has actively established.

In the event a message arrives that could satisfy more than one of these types of NetBIOS Receives, the following hierarchy is observed:

1. Receive (highest priority)
2. Receive-Any-for-a-Specified-Name
3. Receive-Any-for-Any-Name (lowest priority)

Applications issue NetBIOS Send commands to transfer data to the other application. The Send command allows the application to send messages ranging in size from 0 bytes to 64K-minus-1 bytes of data; the data must be in contiguous memory. The application can also issue a NetBIOS Chain Send command that allows data to reside in buffers located in two different storage areas.

With a Chain Send command, data within each of the buffers must be in contiguous memory, though the two buffers themselves do not have to be contiguous. Moreover, each data block can range from 0 bytes up to 64K-minus-one bytes, allowing up to 128K-minus-two bytes to transfer with one Chain Send command.

Sessions are ended by one or both sides issuing a NetBIOS Hang Up command that specifies the LSN of the session to be terminated. The other application is notified of the session termination when it issues a subsequent session command. An application can issue a NetBIOS Session Status command that will indicate the status of a session—existing or cancelled.

General Commands

The NetBIOS general commands provide such NetBIOS services as the following:

- Reset forces the adapter to an initialized state.
- Adapter Status queries a NetBIOS adapter and retrieves operational information such as detected LAN error counts and the adapter's NetBIOS name table.
- Cancel allows applications to cancel commands that have not completed.
- Find Name locates adapters that are using a symbolic name specified in the Find Name command.
- Trace activates a trace of all commands issued to the NetBIOS interface to provide support for diagnostic programs.

Issuing NetBIOS Commands

Applications issue NetBIOS commands by first zeroing out a 64-byte area of memory. This prevents

NEW VERSION 4.5

DYNAMITE DEMOS

Proteus Prototype/Demo System

Why fizzle with an outdated demo program when you can explode with Proteus?

Proteus is the first prototype/demo system to combine complete simulation capability, a powerful Screen Painter, video effects, custom sound and much more in a program you don't have to be a genius to figure out.

And now Proteus supports all graphic modes for the CGA, EGA and VGA in the popular PCX file format.

"A joy to use." . . . Computer Language

■ Complete Simulation Capability
■ Powerful Screen Painter
■ Bit-Mapped Graphics
■ Video Effects
■ Custom Sound

Don't Let Your Next Demo be a Dud!

Find out why software developers from Ashton-Tate to ZyLogic have switched to Proteus.

1-800-634-9986

$199. No royalties. 30-day Money Back Guarantee. MC/VISA/COD/PO accepted.

Helios

P.O. BOX 22869, SEATTLE, WA 98122

Number 258 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

residual data from causing NetBIOS to branch wildly into random memory when the command completes. The application then uses the area to construct a *NetBIOS Control Block*, or *Ncb*. Completing the control block consists of filling various fields that are required by the particular command that will be issued.

Failure to complete the Ncb fields correctly can hang the user's machine, since uncompleted fields are initialized to all zeros. For example, in the case of PC-DOS, if the Ncb specifies a Receive command and the receive data buffer address is inadvertently not specified, the arriving data will obligingly be placed at address 0000:0000, overlaying and corrupting your machine's interrupt vectors. That can be a difficult error to isolate, though its effect is more than somewhat obvious.

After filling in the Ncb, the application then points the ES:BX register pair at the Ncb and issues an INT 5Ch interrupt request. When NetBIOS can report status on the request, it does. However, the particular way it does this varies with the way the particular request was issued.

Listing 1 shows how to issue a NetBIOS interrupt request using the C language. This particular figure issues a NetBIOS interrupt directly to the native NetBIOS interrupt. For IBM PCs, an alternate method uses a PC-DOS INT 2Ah request specifying a value of 0400h or 0401h in the AX register. In both cases the ES:BX register pair point to a valid Ncb. An AX value of 0401h indicates that PC-DOS should not automatically retry the command if the command failed. An AX value of 0400h indicates PC-DOS should retry the command on failure.

The INT 2Ah approach is sometimes necessary for total coexistence with the IBM PC LAN Program. However, before first using this interface, an application must test the version of PC-DOS to verify it is version 3.1 or later, and then determine whether the program is installed. Listing 2 illustrates how to issue a NetBIOS interrupt request using assembly language. This code fragment issues a NetBIOS interrupt directly to the native NetBIOS interrupt in lieu of a PC-DOS INT 2Ah request with AX set to 0400h or 0401h.

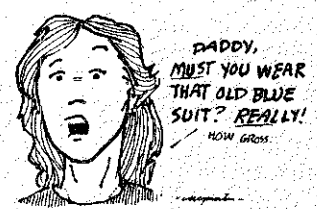
Sample C Test Program

We will close with a look at our first NetBIOS program. PRESENCE.C (Listing 3) is a sample C program that should correctly test for the presence of NetBIOS on all IBM PC models.

The program issues an invalid NetBIOS request to ensure that the NetBIOS interrupt vector is actually being used by NetBIOS and has not been captured by some other program. However, if the interrupt vector has been captured by another program that reflects the request to an uninitialized NetBIOS interrupt, the results may be catastrophic. PJ

W. David Schwaderer is an internationally recognized LAN and data communications authority. He is the author of five data communication programs sold by IBM and of several books, including C Programmer's Guide to NetBIOS *and* IBM's Local Area Networks: Power Networking and Systems Connectivity. *He left IBM after 15 years as a senior programmer to co-found InterFax Inc., the world's first interactive facsimile information retrieval system. His many NetBIOS example programs in his book are available from Sacramento Distribution Service, P.O. Box 3014, Citrus Heights, CA 95611-3014; see his NetBIOS book for details.*

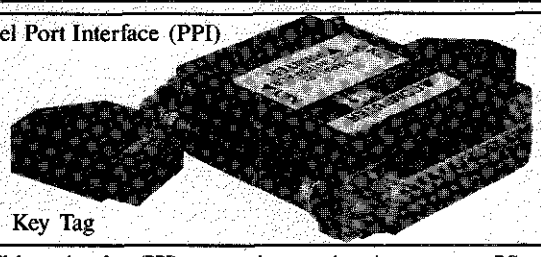
Code follows.



— SERGE SUPPRESSOR —

World Class Software Security

Parallel Port Interface (PPI)



Key Tag

The parallel port interface (PPI) connects between the printer port on a PC and the printer cable. The PPI holds two Key Tags, one on each side. Each Key Tag contains a secure custom chip which is pre-programmed by Glenco to only work with the assigned software package. A second Key Tag can be employed to protect another package, or may be used to turn other software packages "on", remotely or on-site.

- STANDARD KEY TAG - Software is protected for an unlimited number of executions. They are pre-programmed to include a sequentially assigned S/N.
- COUPON KEY TAG - Software is valid for a preset number of executions. The Coupon count can be reset remotely or on the customers site by using a second update Key Tag.
- READ/WRITE KEY TAG - With programmable memory. Perfect for companies which have multiple products or a product with several optional modules. By having several packages protected using one Key Tag, your costs are lowered.
- DURATION KEY TAG - Has a clock on board. (Available late '89)

Secure software and data with reliable, effective protection products that won't burden honest users.

Glenco is a world leader in the area of software security products and services. Our copy protection products and data security products are second to none. They are designed to function on a wide variety of third party hardware. We have over 3500 satisfied software firms utilizing our products. We also have a full line of disk based protection systems.

- MACHINES SUPPORTED - IBM PC/XT/AT & PS/2, Macintosh
- OPERATING SYSTEMS - MS-DOS, XENIX, Network, Finder, & Multifinder.
- LANGUAGES/COMPILER - Over 50, including runtime packages, data bases and spread sheets. We have a non-programmers interface as well.

Call or write for more information.

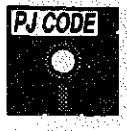
GLENCO ENGINEERING INC.
SERVING THE SOFTWARE INDUSTRY SINCE 1979

721 W. Algonquin Road, Arlington Hts., IL 60005, (312) 364-7638, FAX 364-7698

In Europe contact: SDC Security Systems, The Netherlands
Tel: +31-45-441535, FAX: +31-45-444747

Number **323** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 88.

Code from "Exploiting NetBIOS," by W. D. Schwaderer

1

LISTING 1

```
/* C fragment illustrating a direct NetBIOS interrupt request */

#define NetbiosInt21FunctionCode    (unsigned char) 0x2A
#define NetbiosInt5C                (unsigned char) 0x5C

void NetbiosRequest(NcbPointer)
struct Ncb *NcbPointer;
{
 struct SREGS SegRegs;
 union  REGS  InRegs, OutRegs;  /* defined in dos.h */
 struct Ncb far *NcbPtr = (struct Ncb far *) NcbPointer;

    segread(&SegRegs);

    SegRegs.es  = FP_SEG(NcbPtr);
    InRegs.x.bx = FP_OFF(NcbPtr);

    int86x(NetbiosInt5C, &InRegs, &OutRegs, &SegRegs);
}
```

LISTING 2

```
; Assembly language fragment illustrating a direct netBIOS interrupt request

NetbiosInt    equ 5Ch                    ; NETBIOS interrupt vector
              mov       BX,offset Ncb
              mov       AX,DS            ; our Ncb is in a data segment
              mov       ES,AX            ; ES:BX ==> Ncb
              int       NetbiosInt
```

LISTING 3

```
#define LINT_ARGS

#include <dos.h>
#include <stdio.h>
#include "netbios2.h"          /* the Ncb control block structure */

struct SREGS SegRegs;

#if defined(LINT_ARGS)
extern  int main(int argc,char * *argv);
extern  int CheckDosIVs(void);
extern  void IssueInvalidNetbiosRequest(void);
extern  void ClearNcb(struct Ncb *NcbPtr);
extern  void NetbiosRequest(struct Ncb *NcbPointer);
extern  void Logo(void);
#endif

int main(argc, argv)
int   argc;
char *argv[];
{
   unsigned char temp;

   Logo();

   if (CheckDosIVs())
       IssueInvalidNetbiosRequest();

   printf("\n\nProgram ending...\n");

   return 0;
}
```

2

```
#define DOS_INT_21   0x21
#define DOS_FETCH_IV 0x35

int CheckDosIVs()
{
    union  REGS  InRegs, OutRegs;
    struct SREGS SegRegs;

    InRegs.h.ah = DOS_FETCH_IV;
    InRegs.h.al = NetbiosInt5C;

    int86x(DOS_INT_21, &InRegs, &OutRegs, &SegRegs);

     printf("\n\nNetBIOS Int 5Ch IV SEGMENT:OFFSET == %04X:%04X...",
                                        SegRegs.es, OutRegs.x.bx);

    switch (SegRegs.es) {
      case 0x0000 : printf("\n\nNetBIOS IV segment == 0x0000\x07");
                    return FAILURE;
                    break;
      case 0xF000 : printf("\n\nNetBIOS IV segment == 0xF000\x07");
                    return FAILURE;
                    break;
      default :     printf("\n\nNetBIOS IV segment appears valid...");
                    return SUCCESS;
                    break;
   }
}

#define ERROR_INVALID_COMMAND 0x03

void IssueInvalidNetbiosRequest()
{
   struct Ncb  PresenceNcb;

   ClearNcb(&PresenceNcb);

   PresenceNcb.NcbCommand = NETBIOS_INVALID_COMMAND;

   NetbiosRequest(&PresenceNcb);

   if (PresenceNcb.NcbRetCode == ERROR_INVALID_COMMAND)
      printf("and NetBIOS is present...");
   else
      printf("but NetBIOS is not present...\x07");
}

void ClearNcb(NcbPtr)
struct Ncb *NcbPtr;
{
   int   i;
   char *CharPtr;

   CharPtr = (char *) NcbPtr;

   for ( i = 0; i < sizeof(ZeroNcb); i++ )
      *CharPtr++ = 0x00;
}

void NetbiosRequest(NcbPointer)
struct Ncb *NcbPointer;
{
 union  REGS  InRegs, OutRegs;  /* defined in dos.h */
 struct Ncb far *NcbPtr = (struct Ncb far *) NcbPointer;

    segread(&SegRegs);

    SegRegs.es  = FP_SEG(NcbPtr);
    InRegs.x.bx = FP_OFF(NcbPtr);

    int86x(NetbiosInt5C, &InRegs, &OutRegs, &SegRegs);
}

void Logo()
{
   printf("\n*- NetBIOS Presence Test Program");
   printf(" (C) Copyright 1988 W. David Schwaderer -*");
}
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Dynamic Data Exchange in Windows

by
William S. Hall

Dynamic data exchange, or DDE, is a protocol for real-time data transfer among cooperating Windows programs. It is one of several means of moving data from one Windows application to another. In this first of two installments, we will examine the main elements of DDE; in the second, we will look at a simple communications program.

The transfer of data from one Windows application to another can be accomplished in many ways. The most common way is via the Windows Clipboard. Many Windows programs provide a means for the user to select all or part of the client area and copy it to the Clipboard. Its contents can then be transferred back to the same application or into another. The Clipboard, an example of a *static* method of data transfer, requires user intervention.

In *PJ* 7.2, we showed that the Windows message system can provide a simple *dynamic* link between two programs. *[See references at the end of the article. — Ed.]* In this example, debugging information in the Windows program is formatted as a string, and its length and pointer are sent via a WM_USER message to another, called WINAUX, where it is displayed. Initial contact is made by retrieving the Window handle of WINAUX from WIN.INI. WINAUX is an example of using private messages to effect a one-way transfer of text data.

DDE formalizes and extends the simple ideas behind WINAUX into a series of transactions from which a DDE session can be constructed. In addition, DDE allows data in a variety of formats to be exchanged. Nine Windows messages have been defined to support the protocol, and ingenious ways have been devised to transport the data within the limits of the Windows message parameters *wParam* (a word) and *lParam* (a long integer).

Although the official documentation from Microsoft advertises DDE as easy to implement, it is more complicated than it seems. In addition, a degree of vagueness exists about what constitutes success or failure of a transaction. This is important to know, because many exchanges involve the creation of global objects that must be destroyed, sometimes by the sender and at other times by the receiver. The failure of a transaction can lead to an accumulation of these objects in the global heap with no application responsible for their removal. (In the second installment, we will look at how we interpret the specification when we study the communications server–client application.)

A number of interesting examples on DDE have appeared in *Microsoft Systems Journal* (see references). The code for these examples can be downloaded from Microsoft's ONLINE facility (and possibly from other bulletin boards).

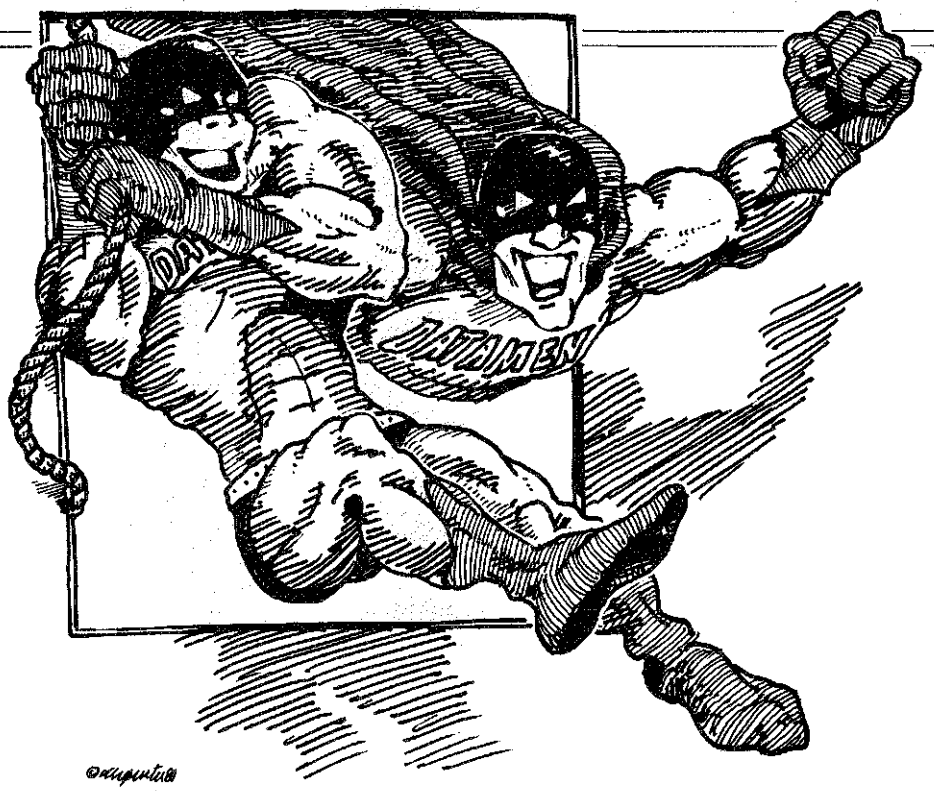
General Principles of DDE

DDE's principal use is to provide data links that function automatically and with a minimum of user intervention. One of the more striking examples uses Lotus Signal: The server obtains the information from Signal and provides it via DDE to a client application, such as an Excel spreadsheet, that is maintaining a graphic or numeric display of current activities regarding certain stocks. Harvey Berger's article in *Microsoft Systems Journal* illustrates a program where the Signal hardware is simulated by the server and the client displays the results as a bar graph.

A Windows program can act as a server to one application and a client to another. Even in the middle of a session on one topic, the server can initiate a session with the client on another, effectively reversing their roles on the new topic. More generally, a single Windows program can be acting as a client or server (or both) to any number of other Windows applications. Kevin P. Welch's article in *Microsoft Systems Journal* demonstrates the versatility of DDE with a complex program that swaps a moving ball from application to application using DDE to control the exchange.

A DDE session is partly identified by the window handles of the client and the server. Normally, a unique client window is associated with each session; a single server window, on the other hand, may be able to handle several clients. A client window does not have to be the main window of the application. It is quite possible for a child window of an application to be a DDE client. One can imagine, for example, a spreadsheet program where each spreadsheet is engaged in a separate DDE conversation.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



To accomplish its tasks, DDE defines a hierarchy to identify uniquely a unit of data in terms of item, topic, and application. An *item* is a data object that can be passed in a DDE exchange. It could be a single character, a string, or a bit map. A *topic* is a logical data context—a filename, for example. Finally, the *application* identifies the server capable of executing the client's wishes.

In the communications example that we will use in the next installment, the application is the name of the server, the topics are the names of the communications ports (COM1, COM2, etc.), and the items are the strings "Get" and "Send". In the Lotus signal example briefly described above, the application is "SIGNAL," the topic is "NYSE" (for New York Stock Exchange), and an item is the standard abbreviation for a company carried on the exchange.

To carry out the data transfer, DDE uses a set of messages that can be combined to form transactions. These can be assembled in turn by the programmer into a complete DDE session. Again, referring to the forthcoming communications example, a typical session begins by **initiating** a connection to the server on a selected port. If the port is available, an **execute** command is sent to the server to set baud rate, parity, and byte size. Then the server is **advised** to supply data from the communications port for display by the client whenever new information is available. At any time, the client may **poke** keystrokes from the client to the server for transmittal. Additionally, the client may **unadvise** the server in order to suspend incoming information. The session ends when the client **terminates** the conversation.

DDE Messages

WM_DDE_INITIATE	Begin a DDE session.
WM_DDE_TERMINATE	End a DDE session.
WM_DDE_EXECUTE	Execute a command in the server.
WM_DDE_ACK	Acknowledge a DDE message.
WM_DDE_POKE	Send a data item to the server.
WM_DDE_REQUEST	Request a data item from the server.
WM_DDE_ADVISE	Establish a data link with the server.
WM_DDE_UNADVISE	Terminate a data link with the server.
WM_DDE_DATA	Supply data to client as required by the **ADVISE** or **REQUEST** message.

DDE Message Format

A typical DDE message has the following form:

```
PostMessage(hWndReceiver, WM_DDE_XXX, hWndSender,
            MAKELONG(val1, val2));
```

SendMessage replaces **PostMessage** only during the initiation and corresponding acknowledgment of a DDE session. The window handle of the target, *hWndReceiver*, is acquired when the potential client begins a session; *hWndSender* is, of course, the sender's window handle. **WM_DDE_XXX** is one of the nine defined DDE messages (which we shall examine later).

The last parameter in the message is a 32-bit quantity that is a composite of various kinds of data objects. These may be atoms, status words, handles to global data, and registered Clipboard formats. Let's look at the nature of each.

Atoms

By now, you are beginning to suspect that the application, topic, and item are represented by strings. Strings are clumsy to pass between applications, since they can be identified uniquely only by long pointers. However, Windows provides a method of associating a string with a 16-bit datum called an *atom*, which can be either local or global. A local atom cannot be seen by other applications. A global atom, which can be read by all, is the mechanism for encoding the name of the application and the strings representing the topic and item.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

DDE MESSAGES

Message:	**WM_DDE_INITIATE**
Purpose:	To request initiation of a DDE conversation.
Sender:	Client
Receiver:	Server
Format:	SendMessage((HWND)(-1), WM_DDE_INITIATE, hWndClient, MAKELONG(aApplication, aTopic))
aApplication:	Atom specifying name of server or null.
aTopic:	Atom specifying topic or null.
Sender Tasks:	Atoms are added before sending message and removed upon return.
Receiver Tasks:	Applications that can support *aTopic* and whose names match *aApplication* are expected to respond with **WM_DDE_ACK**. A null value of *aApplication* matches any application. A null value of *aTopic* matches any topic and requires a separate response for each one supported.
Remarks:	Application names must not contain slashes or backslashes.

Message:	**WM_DDE_TERMINATE**
Purpose:	To end a DDE conversation.
Sender:	Client/Server
Receiver:	Server/Client
Format:	PostMessage(hWndClient/hWndServer, WM_DDE_TERMINATE, hWndServer/hWndClient, 0L)
Receiver Task:	The recipient is expected to reply with a **WM_DDE_TERMINATE** message.
Remarks:	Sender promises to send no more messages to receiver. Receiver has permission to close its window.

Message:	**WM_DDE_REQUEST**
Purpose:	To request a data item from the server.
Sender:	Client
Receiver:	Server
Format:	PostMessage(hWndServer, WM_DDE_REQUEST, hWndClient, MAKELONG(cfFormat, aItem))
cfFormat:	A registered clipboard format word.
aItem:	Atom specifying item.
Sender Tasks:	Atom is added before posting message.
Receiver Tasks:	If server has access to *aItem* and can render it in the correct format, a **WM_DDE_DATA** message is sent; otherwise, a negative **WM_DDE_ACK** message is returned. The *aItem* atom can be reused or can be deleted and re-added by the subsequent **WM_DDE_DATA** or **WM_DDE_ACK** message.
Remarks:	A client may ask for the same item repeatedly using different formats until it finds one the server can support.

Continued . . .

When a string is passed from one application to another, it is entered into the global atom table. The atom returned is then sent to the application via a Windows message, and the corresponding string is then retrieved by the recipient.

The following are the function calls for entering and retrieving strings from atom tables:

GlobalAddAtom	Create a global atom from a string.
GlobalDeleteAtom	Delete a global atom if reference count is zero.
GlobalFindAtom	Retrieve an atom associated with a given string.
GlobalGetAtomName	Copy the character string associated with a global atom.

Local atoms have corresponding function calls. Both kinds are useful when using DDE. The local functions can be called at program initialization to store a table of atoms that are to be used during a session. As each is needed, the local-atom string can be retrieved and then added to the global heap. The returned global atom is then passed by the message system to the recipient.

Atoms must be managed carefully to avoid their accumulation in memory. However, the general principle is fairly simple: the sender adds the atoms, and the receiver removes them. There are some exceptions, of course. For example, if a client posts a message containing atoms to a server and an acknowledgment is expected, the server can simply reuse the atoms for the **WM_DDE_ACK** reply. The client will then remove the atoms when the acknowledgement is received. Atoms are also handled in a special way during initial contact, where the atoms are added and removed by the sender.

Global Data

DDE uses shared memory objects to carry data, command strings, or option information. They are always treated as read only and cannot be used as mutual read/write areas.

The sender allocates a DDE shared object with the command

```
hData = GlobalAlloc(GMEM_MOVEABLE | GMEM_DDESHARE,
            dwBytes);
```

The handle returned by **GlobalAlloc** is locked to obtain a long pointer, and the appropriate data is added. The handle is then unlocked and passed as a part of *lParam* in a DDE message. The recipient uses the data in a similar way by locking the handle, reading the data, and then unlocking the handle.

Each type of message that passes a data handle assembles the information in a unique way. Usually, the first word contains flags indicating the disposition of the handle after use. The second word contains a Clipboard format number describing the type of the data that follows. The data itself then follows. An exception is an execution string, where there are no preliminary fields.

As with atoms, there are rules for managing the data. Usually the recipient frees the object; however, if the receiver is busy, it may return an indication of such, letting the responsibility fall back upon the sender. In some cases, the sender normally removes the object. An example is a command string in the **WM_DDE_EXECUTE** message where the recipient simply returns the handle in the acknowledgment message to the sender, which removes the object. Finally, the sender may tell the receiver explicitly not to free the object.

Status Word

The third type of data item that can be used in *lParam* is a status word. These appear in some **WM_DDE_ACK** messages. Bit fields indicate the result to the client of a previously sent DDE message. Bit 15 shows whether or not the request was accepted; bit 14 signals that the server was busy. Finally, the low byte can be used for application-dependent information.

Clipboard Formats

Windows understands a number of standard data formats. The most common is CF_TEXT, which is simply text in which each line is terminated by a carriage return–line feed pair and the end of the data is signaled

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

by a null character. Other formats include bit maps, metafiles, and a variety of private data types that can be registered by an application. In DDE, the server and client must agree on the data formats to be used before data can be exchanged.

DDE Transactions

Now let us look at each of the transactions that are available in DDE. The goal of this section is to indicate the behavior of the client and the server until the transaction is completed. Details of the messages themselves and their associated data structures can be seen in the panels.

Initiate a Conversation

CLIENT: To begin a DDE session, a potential client broadcasts a message to all other applications in the following form:

```
SendMessage((HWND)-1, WM_DDE_INITIATE, hWndClient,
            MAKELONG(aApplication, aTopic));
```

where either of the atoms *aApplication* or *aTopic* can be null. The **SendMessage** call does not return until all applications able to respond have sent acknowledgment messages.

SERVER: A potential server examines the atoms in the **WM_DDE_INITIATE** message received. If *aApplication* matches the server's name or is null and if *aTopic* can be supported, the server responds with a reply in the following form:

```
SendMessage(hWndClient, WM_DDE_ACK, hWndServer
            MAKELONG(aApplication, aTopic));
```

If the sender's *aTopic* is null, a message is sent for each topic the server can manage.

CLIENT: If there are servers that can support the topic, one or more **WM_DDE_ACK** messages will be received while waiting for the initial **SendMessage** call to return. For each acknowledgment received, the client updates appropriate data structures for each conversation accepted and terminates those that are unwanted.

After all acknowledgments (if any) have been received, the original **SendMessage** call returns to the client. The client may now begin other DDE transactions with the server on the agreed-upon topic.

Terminate a DDE Session

SENDER: Either the client or the server may terminate a DDE session at any time with

```
PostMessage(hWndRecipient, WM_DDE_TERMINATE,
            hWndSender, 0L);
```

The sender signals with this message that no more messages will be sent and implies that the receiver may close its window. If the sender receives any other messages than **WM_DDE_TERMINATE** from the receiver, no response must be sent, since at this point the receiver's window may have been destroyed.

RECEIVER: The receiver replies promptly with the same message.

Message:	**WM_DDE_DATA**
Purpose:	To send a data item to the server.
Sender:	Server
Receiver:	Client
Format:	PostMessage(*hWndClient*, WM_DDE_DATA, hWndServer, MAKELONG(hData, altem))
hData:	typedef struct { *unsigned unused:12*, fAck:1, fRelease:1, fReserved:1, fAckReq:1; int cfFormat; BYTE rgb[1]; } DDEUP;
altem:	Atom specifying item.
Sender Tasks:	Atom is added before posting message unless reused from the preceding **REQUEST** message. Set *fAck* to 1 if receiver is expected to **ACK** message. Set *fRelease* to 1 if receiver is to free data item. Set *fAckReq* to 1 if offered in response to a **WM_DDE_REQUEST** message. Set *cfFormat* to format of data. Supply data in *rgb* field.
Receiver Tasks:	Send positive **ACK** to sender (if requested) after successfully processing data. Free *hData* if required. Reply with negative **ACK** if operation fails. Do not free *hData* regardless of *fRelease*. If no **ACK** required, delete the atom; otherwise, the *altem* atom can be reused or can be deleted and re-added by the subsequent **WM_DDE_ACK** message.

Message:	**WM_DDE_ADVISE**
Purpose:	To request a data link with the server.
Sender:	Client
Receiver:	Server
Format:	PostMessage(hWndServer, WM_DDE_ADVISE, hWndClient, MAKELONG(hOptions, altem))
hOptions:	typedef struct { unsigned unused:13, *fRelease*:1, fDeferUpd:1, fAckReq:1; int cfFormat; } DDELN;
altem:	Atom specifying item.
Sender Tasks:	Atom is added before posting message. *fAckReq* set to 1 if client wants server to send **WM_DDE_DATA** messages with **ACK** requested bit set in order to obtain a degree of flow control. *fDeferUpd* set to 1 if client wants an alarm that data is present but not the actual data. The data can then be obtained by a

Continued . . .

Request Data Item from Server

CLIENT: The client sends the server a message in the following form:

```
PostMessage(hWndServer, WM_DDE_REQUEST, hWndClient,
            MAKELONG(cfFormat, aItem));
```

where *aItem* is an atom identifying the desired data item and *cfFormat* is a registered Clipboard format recognized by Windows.

SERVER: If the server can supply the data in the required format, a reply is sent as

```
PostMessage(hWndClient, WM_DDE_DATA, hWndServer,
            MAKELONG(hData, aItem));
```

where *aItem* describes the item and *hData* is a handle to a shared data block. The first word of the data block contains flags indicating the disposition of the data, and the second holds the format type. The remaining bytes hold the actual data.

If the server is busy or the data cannot be rendered in the desired format, the server replies with

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

	WM_DDE_REQUEST message, or the alarm can be ignored.
	fRelease is set to 1 if *hOptions* is to be freed by the recipient.
	clFormat is set to the required clipboard format.
Receiver Tasks:	Send positive **ACK** to sender if the data for *aItem* can be provided in the requested format. Free *hOptions* if *fRelease* is set to 1.
	Reply with negative **ACK** if unable.
	The *aItem* atom can be reused or can be deleted and re-added by the subsequent **WM_DDE_ACK** message.
Remarks:	The server sends new data to client via the **WM_DDE_DATA** message.

Message:	**WM_DDE_UNADVISE**
Purpose:	To terminate data updated by server.
Sender:	Client
Receiver:	Server
Format:	PostMessage(hWndServer, WM_DDE_UNADVISE, hWndClient, MAKELONG(NULL, aItem))
aItem:	Atom specifying item, which can be null.
Sender Tasks:	Atom is added before posting message.
Receiver Tasks:	Send positive **ACK** to sender if the request can be honored; otherwise, reply with a **NAK**.
	The *aItem* atom can be reused or can be deleted and re-added by the subsequent **WM_DDE_ACK** message.
Remarks:	If *aItem* is null, all **ADVISE** transactions with the client must be terminated.

Message:	**WM_DDE_POKE**
Purpose:	To send a data item to the server.
Sender:	Client
Receiver:	Server
Format:	PostMessage(hWndServer, WM_DDE_POKE, hWndClient, MAKELONG(hData, aItem))
hData:	typedef struct { unsigned unused:12, fRelease:1, fReserved:2; int clFormat; BYTE rgb[1]; } DDEPOKE;
aItem:	Atom specifying item.
Sender Tasks:	Atom is added before posting message.
	fRelease is set to 1 if *hData* is to be freed by the server.
	clFormat is set to the required clipboard format.
	rgb contains the data, which may be of any size.
Receiver Tasks:	If operation is successful, reply with positive **ACK**. If *fRelease* is 1, free *Data*.
	If operation fails, reply with negative **ACK**. Do not free *hData*.

Continued . . .

```
PostMessage(hWndClient, WM_DDE_ACK, hWndServer,
            MAKELONG(wStatus, aItem));
```

where wStatus contains flags describing the nature of the failure.

CLIENT: Upon receipt of a **WM_DDE_DATA** message, the client processes the data as required. The data handle is freed according to the flags in the first word of the data block. If an acknowledgment was requested, it is returned to the server as

```
PostMessage(hWndServer, WM_DDE_ACK, hWndClient,
            MAKELONG(wStatus, aItem));
```

where *wStatus* contains flags describing the results.

If a **WM_DDE_ACK** is received, it is because the request failed. In this event, the client may begin the request process again by substituting a different Clipboard format.

Send Data Item to Server

CLIENT: The client sends the server a message in the following form:

```
PostMessage(hWndServer, WM_DDE_POKE, hWndClient,
            MAKELONG(hData, aItem));
```

The desired item is identified by *aItem*. *hData* is a handle to a shared memory object whose first word contains flags indicating the disposition of the object by the server. The second word contains the data format, and the remaining bytes contain the data.

SERVER: If the server can accept the data in the rendered format, it processes the data. Whether or not it accepts, the server returns a message in the following form:

```
PostMessage(hWndClient, WM_DDE_ACK, hWndServer,
            MAKELONG(wStatus, aItem));
```

where *aItem* describes the item and *wStatus* contains flags describing the success or failure of the transaction.

Establishing a Data Link

CLIENT: The client sends the server a message in the following form:

```
PostMessage(hWndServer, WM_DDE_ADVISE, hWndClient,
            MAKELONG(hOptions, aItem));
```

The desired item is identified by *aItem*, and *hOptions* is a handle to a shared memory object whose first word contains option flags and whose second word contains the desired Clipboard data format.

Once the link has been established, the server is supposed to supply the client with new data as it becomes available via the data-update sequence. However, the options flags can be used to regulate the data flow. For example, the client can ask the server to send the next packet of data only after an acknowledgment message has been received from the client for the previous item. And, if the client desires, the server can be requested to send, instead of the data itself, only an alarm that the data has changed. In this case, the client can then pick up the new information with a **REQUEST** command.

SERVER: If the server can provide the information in the requested format, it prepares itself to supply the data whenever it changes to the client. Whether or not it can supply the data, the server returns a message in the following form:

```
PostMessage(hWndClient, WM_DDE_ACK, hWndServer,
            MAKELONG(wStatus, aItem));
```

where wStatus contains the disposition of the request.

Data Update

SERVER: The server sends the client a message in the following form:

```
PostMessage(hWndClient, WM_DDE_DATA, hWndServer,
            MAKELONG(hData, aItem));
```

The desired item is identified by *aItem*, and *hData* is a global handle to a shared memory object whose first word contains flags indicating the disposition of the object by the server and whose second word contains the data format. The remaining bytes contain the data.

CLIENT: The client processes the data and, if requested to do so, releases the data handle according to bits in the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The C Programmer's Companion
Now Available for OS/2 !!!

CCSYM - C Symbol Compiler

* Analyzes ANSI C source code. Microsoft/Turbo extensions supported.
* Automatically generates separate commented function prototype files for all global and local (static) functions.
* Log File lists all your symbol definitions with detailed usage statistics.
* Generates a "symbol database" which CCRIDER uses to quickly retrieve ANY symbol definition in your application, no matter how large it is.
* Incremental compilation supports efficient database updates.

CCRIDER - Pop-up Symbol Lookup

* Pop-up utility which converts your own editor into a powerful hypertext programming environment.
* Point at a symbol in your editor, then pop-up its definition in a window.
* One keystroke "Edit" command allows you to edit the definition in your own editor, no matter which file or directory it was defined in.
* "Paste" command inserts the definition into your currently edited file.
* Supports multiple editing sessions under OS/2.

DOS Version $89
OS/2 Version $179

FREE DEMO

Western Wares
Box C Norwood, CO 81423
(303) 327-4898

Number **100** on the Reader Service Card

Introducing C talk / Views™

Reduce C Programming for MS Windows by 75%

Introducing C_talk/Views™, a major productivity tool for Microsoft® Windows program development in C. Slash your MS Windows program development time by up to 75% through the powerfull object-oriented programming (OOP) technology of C_talk. With C_talk/Views, the most common elements of all MS Windows programs are packaged into reusable software components-object classes. Over forty *software components are included as a* complete applications development framework. These object classes provide an easy-to-learn and powerfully consistent interface to the rich functionality of MS Windows. C_talk/Views is also a real OOP environment . Use the software components as they are, or extend them to meet the most demanding applications.

Coming soon...C_talk/Views for the Apple Macintosh®... now your MS Windows program will port to the Mac with little or no change!

$450.00 In the US, add $5 for shipping. *($25 for int'l handling)*

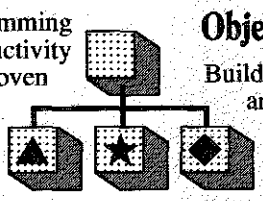
CNS, Inc.
Software Products
7090 Shady Oak Rd.
Minneapolis, MN 55344
612-944-0170, Fax 612-944-0923

. . . providing and advancing object-oriented methodology.

Contact this advertiser directly.

The Professional Library for Object-Oriented Pascal

Object-oriented programming multiplies your productivity by promoting reuse of proven software modules. Don't start from scratch! Use **Object Professional 1.0**, a powerful library of over 30 object types containing over 1000 methods.



Powerful User Interfaces

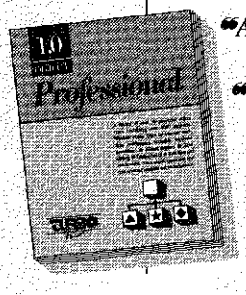
The window object types let you use multiple overlapping and resizeable windows. The windows provide capability for ■ mouse support ■ scroll bars ■ menus ■ text editing ■ dialog boxes ■ pick lists ■ scrolling data entry screens ■ printed forms ■ help capability, and more. The window classes are incredibly flexible — you can create text-mode PM look-alikes or your own unique look and feel.

Object Oriented Data Too

Build your programs from proven and documented object types like stacks, linked lists, virtual arrays, and more. Make your own custom data types by simply inheriting from one of the provided types and adding your own methods and instance variables.

The Impossible, Made Easy

System-oriented routines provide ■ swappable TSRs in only 6K of RAM ■ DOS and BIOS capabilities ■ EMS management ■ keyboard macros ■ interrupt management ■ swapping Exec manager, and much more.



Complete Documentation, Full Source

Object Professional has three volumes of complete documentation, online reference guide, free technical support, and full source code. You pay no royalties. Hot demo programs show you how to use the power of Object Professional. You'll get up to speed fast with OOP!

Object Professional is the successor to the acclaimed Turbo Professional:

"A superbly crafted toolbox."
Kent Porter, DDJ, 4/88

"The range of this toolkit is simply astonishing."
Jeff Duntemann, DDJ, 5/89

Object Professional 1.0, only $150.

Call toll-free to order.

1-800-333-4160

8AM - 5PM PST Monday through Friday, USA & Canada.
For more information call (408) 438-8608.
TurboPower Software PO Box 66747 Scotts Valley, CA 95066-0747

Satisfaction guaranteed or your money back within 30 days. Turbo Pascal 5.5 is required. Add $5 for shipping in U.S. and Canada. Elsewhere add $35 per unit. T.PRO customers may purchase O.PRO for $100 plus shipping. Include your serial number.

TURBO Power

Number **25** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Message:	**WM_DDE_EXECUTE**
Purpose:	To execute a command in the server.
Sender:	Client
Receiver:	Server
Format:	PostMessage(hWndServer, WM_DDE_EXECUTE, hWndClient, MAKELONG(NULL, hCommand))
hCommand:	Global memory item containing execution string.
Sender Tasks:	Sender should render the command using the correct syntax.
Receiver Tasks:	Execute the commmand and return a positive **ACK** if successful; otherwise, return a negative **ACK**. Receiver does not free *hCommand*.

Message:	**WM_DDE_ACK** of **WM_DDE_INITIATE**
Purpose:	To respond to **WM_DDE_INITIATE**.
Sender:	Server
Receiver:	Client
Format:	SendMessage(hWndClient WM_DDE_ACK, hWndServer, MAKELONG(aApplication, aTopic))
aApplication:	Atom specifying name of server.
aTopic:	Atom specifying topic.
Sender Tasks:	Atoms are added before sending message.
Receiver Tasks:	Atoms are removed upon receipt.
Remarks:	Application names must not contain slashes or backslashes. Client must terminate all unwanted conversations.

Message:	**WM_DDE_ACK** of **WM_DDE_EXECUTE**
Purpose:	To respond to **WM_DDE_EXECUTE**.
Sender:	Server
Receiver:	Client
Format:	SendMessage(hWndClient WM_DDE_ACK hWndServer, MAKELONG(wStatus, hCommand))
wStatus:	Data format word. Bit 15 is set to show success. Bit 14 is set if client is busy. Bits 7 to 0 are application specific. Other bits are reserved.
hCommand:	Handle to command string received from client.
Receiver Tasks:	Free *hCommand*.

Message:	**WM_DDE_ACK** of other messages
Purpose:	To inform client/server of success or failure of transaction.
Sender:	Server/Client
Receiver:	Client/Server
Format:	PostMessage(hWndClient/hWndServer, WM_DDE_ACK, hWndServer/hWndServer, MAKELONG(wStatus, aItem))
wStatus:	Data format word. Bit 15 is set to show success. Bit 14 is set if client is busy. Bits 7 to 0 are application specific. Other bits are reserved.
aItem:	Atom specifying item.
Sender Tasks:	Atom from previous message can be reused, or it can be removed and re-added.
Receiver Tasks:	*aItem* is removed. Free any data handle if **ACK** is in response to a message in which the recipient was not requested to free the object or if **ACK** is negative or busy.

flag word. If an acknowledgment is requested, a message is returned in the following form:

```
PostMessage(hWndClient, WM_DDE_ACK, hWndServer,
            MAKELONG(wStatus, aItem));
```

where *aItem* describes the item and *wStatus* contains flags describing the success or failure of the transaction.

Terminating a Data Link

CLIENT: The client sends the server a message in the following form:

```
PostMessage(hWndServer, WM_DDE_UNADVISE, hWndClient,
            MAKELONG(0, aItem));
```

The item to terminate is identified by *aItem*. If *aItem* is null, all data links associated with the client are terminated. Note that this message signifies termination of only the data link—not of the session.

SERVER: If the termination message can be processed, the server responds with

```
PostMessage(hWndClient, WM_DDE_ACK, hWndServer,
            MAKELONG(wStatus, aItem));
```

where, as usual, *aItem* identifies the item and *wStatus* contains the result of the request.

Executing a Command

CLIENT: To execute a command in the server, the client sends

```
PostMessage(hWndServer, WM_DDE_EXECUTE, hWndClient,
            MAKELONG(NULL, hCommand));
```

where *hCommand* is a global data object containing the command string. The format of the command follows a special syntax, which is described in the next section of this article.

SERVER: The server responds with

```
PostMessage(hWndClient, WM_DDE_ACK, hWndServer,
            MAKELONG(wStatus, hCommand));
```

where *wStatus* shows the success or failure of the command. The client is expected to release *hCommand*.

Syntax of the Execute Command String

An execution string in DDE consists of any number of commands contained in brackets, that is, **[*command1*][*command2*]....** The command strings themselves consist of an opcode and an optional parameter list in parentheses.

Using the notation of yacc, a DDE execution string should follow these rules:

```
exestring : exestring command
          | /* nothing */

command : '[' opcodestr ']'
;

opcodestr : opcode
          | opcode parameterlist
;

parameterlist : '(' parameters ')'
;

parameters : parameter
           | parameters ',' parameter
;

parameter : operand
          | '"' operand '"'
;
```

Both *opcode* and *operand* are application defined. The former must not contain spaces, commas, parentheses, or quotes. The latter must not include commas or parentheses except inside quoted strings.

In the example in the next installment, we will set the communications port from the client with an execute string of the form **[setdcb("b,p,s")]**, where *b* is the baud rate, *p* is the parity, and *s* is the word size. To make it interesting, the string will be parsed by yacc, and we shall show one way to use yacc and lex in Windows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C THE LIMIT!

MicroWay is your best source for the software and hardware you need to get true 32-bit performance from your 386. Our NDP C compiler takes the original C concept of writing lower level code with a higher level language to the limit by providing an inline assembly language interface that lets the programmer specify which register is used to store a variable. This feature makes it possible to use ports or perform interrupts or block moves inline, instead of through calls. The use of register aliased variables to control hardware reduces the size of critical code sequences by a factor of 3 to 10 and keeps the 386's pipelines full by eliminating costly calls. If you are interfacing DOS or the ROM BIOS, or writing graphics routines, a device driver, operating system kernel, or an embedded application, you owe it to yourself to try NDP C-386.

Our Users Report:

Milt Capsimolis of Ithaca Software in Ithaca, NY, developer of HOOPS, the highly regarded 3D, object-oriented graphics library, reports, "We ported a huge library — well over 100,000 lines — without a hitch, in less than a day! NDP C-386 was the fastest 386 compiler we used...We also liked the enormous advantage it offers through its support of the Weitek coprocessor."

Fred Ziegler of AspenTech in Cambridge, Mass. reports, "I ported 900,000 lines of FORTRAN source in two weeks without a single problem!" AspenTech's Chemical Modeling System is in use on mainframes worldwide and is probably the largest application to ever run on an Intel processor.

Please call (508) 746-7341 for more information.

NDP C is also the language of choice if you are combining C with FORTRAN or Pascal, are planning to use any one of four coprocessors that run with the 80386, or require the highest globally optimized code attainable. MicroWay's C, FORTRAN and Pascal compilers come with a 70 function, device-independent graphics library that automatically supports Monochrome, Hercules, CGA, EGA, and VGA adapters and makes it easy to interface memory-mapped peripherals such as digitizers or serial devices such as mice. We make it possible for you to write your own numeric exception handler and include examples written in C, FORTRAN and Pascal. Finally, our C is not only one of the easiest to use, but supports two dialects: ANSI and UNIX. In fact, our UNIX implementation is so close to the standard, that our best customer has become AT&T!

386 Compilers and Tools

NDP Fortran-386™, NDP C-386™, and **NDP Pascal-386™** compilers generate globally optimized, mainframe quality code that runs on the 386 or 486 in protected mode under UNIX, XENIX or Phar Lap extended DOS. The compilers address 4 gigabytes of memory while supporting the 80287, 80387 and Weitek coprocessors. They all come with a library of over 70 device-independent graphics, keyboard and sound routines. Applications can mix code from all three compilers and assembly language. The DOS versions allow the user to write his own numeric error handlers and interface 386 real mode programs from protected mode. The VM versions use Phar Lap's Virtual Memory Manager to run programs which exceed the size of your system memory. **NDP Fortran-386** is a full FORTRAN 77 with FORTRAN 66, BSD 4.2, DOD and VMS extensions. **NDP C-386** is a full K&R C with both MS and ANSI extensions. It is 100% compatible with UNIX C and is substantially faster than the C which comes with UNIX. **NDP Pascal-386** is a full ANSI/IEEE Pascal, with extensions from C and BSD 4.2 Pascal.

DOS versions (require Phar Lap Tools) . $595
VM version (requires VMM) $695
UNIX/XENIX versions $795
Phar Lap Development Tools $495
Phar Lap Memory Manager (VMM) . . . $295

NDPWindows™ — 80 functions which create, store, and recall menus and windows. Works with NDP C and drives all popular graphics adapters . . . . . Library: $125, C Source: $250

NDP HOOPS™ — An NDP port of Ithaca Software's HOOPS, this 3D object-oriented graphics library, callable from NDP C, makes it possible to develop full-featured 386 CAD packages and front ends $575

NDP Plot™ — Calcomp compatible plot package, callable from NDP Fortran. Includes drivers for popular plotters and printers. Works with MDA, CGA, EGA and VGA $325

NDP/FFT™ — The fastest running FFTs on a PC! 40 hand-coded routines that handle 1 and 2 dimensional data arrays. Includes an in core solver that spills to disk for arrays too large to fit in memory.

8088/87/286/287/386/387 version $250
Protected mode NDP version $250

NEW! C++

NDP C++ is a MicroWay port of the UNIX C++ preprocessor version 1.2. It runs in protected mode on DOS, UNIX or XENIX, and is ideal for writing numerics and graphics applications. The product comes with an example of how to support complex numbers in C++ $495

Parallel Processing

Videoputer™ — The highest performance graphics card on the market. Uses a T800 and TI 34010 in conjunction with an 80 MHz Brook-Tree DAC. . . . . With one meg $4495

Monoputer™— Includes one T800 and up to 16 meg of RAM for parallel code development. The four MWhetstones T800 makes this the ideal FORTRAN engine for cost-effective execution of your mainframe programs. $1295

Quadputer™ — This board for the AT or 386 can be purchased with 1 to 4 transputers and 1 or 4 meg of memory per transputer. Two or more Quadputers can be linked together to build networks of up to 100 or more transputers providing mainframe power . . . . . from $1995

Transputer Compilers and Applications
These parallel languages are designed for use with either a Monoputer or Quadputer.

Logical Systems Parallel C $595
3L Parallel C, FORTRAN, or Pascal . $895
ParaSoft EXPRESS Package:— Includes transputer communications libraries, C source level debugger and system Performance Monitor. $1500
COSMOS/M — Finite element analysis. Comes bundled with a Quadputer. Performs at near supercomputer speeds . . . . . from $5995
T800/NAG™ — Features a library of 268 engineering and scientific numerical algorithms. Callable from 3L Fortran. $2750

NDP to HALO '88 Graphics Interface — Lets you call HALO '88 from NDP compilers . . . $100

387BASIC™ — Our 16-bit MS compatible compiler introduces numeric register variables to produce the fastest running 80x87 code on the market $99

386 Your AT

386/387Turbo-AT™ — This board plugs into the 80286 socket, allowing your IBM AT to run 32-bit protected mode code written for the 80386. Includes an 80387 socket. The most cost-effective AT upgrade!

386/387Turbo- AT 16MHz $495
386/387Turbo-AT 25 MHz $695

Weitek-Based Coprocessor Boards

mW1167™ and **mW3167™** coprocessor boards are built at MicroWay using Weitek components. Each includes an 80387 socket.

mW1167-16 . . . . . $895
mW1167-20 . . . . . $1095
mW1167 Microchannel-16/20 . . from $995
mW3167 Microchannel-25/33 . . . from $1795
3167-20 . . . . . $995
3167-25 . . . . . $1295
3167-33 . . . . . $1695
mW3167/80387 Board $150

Intel Coprocessors and RAM

All coprocessors include our 87Test program.

8087	$84	**80287-2**	$120
80287-8	$195	**80287-10**	$220
80387-16	$350	**80387-16SX**	$310
80387-20	$400	**80387-25**	$500

80C287A (CMOS) $280
287Turbo-12 (for AT compatibles) $350
RAMpak™- one meg 32-bit memory module for Compaq 386 20/25 . . . . . $395
RAMpak™- four meg module $1495
Please call for our 100 and 80 ns RAM prices.

Multi-User Accelerators

MicroWay's **AT8™** and **AT16™** intelligent serial controllers run 8 to 16 terminals under UNIX or XENIX without bogging down your AT, 80386 or PS/2. **AT8**: $895 **AT16**: $1295

12 MHz PC Accelerators

SuperCACHE-286 12 MHz $399
FastCACHE-286 12 MHz $299
FastCACHE-286 9 MHz $199

NUMERICS HOTLINE
(508) 746-7341

MicroWay

World Leader in PC Numerics

Corporate Headquarters: P.O. Box 79, Kingston, MA 02364 USA (508) 746-7341
32 High St., Kingston-Upon-Thames, U.K., 01-541-5466
USA FAX 508-746-4678 Italy 02-74.90.749 Holland 40 836455 Germany 069-75-2023

Number **104** on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

4c™ Hypertext Source Code Databank for Programmers

Explore source code! Navigate uncharted logic trails easily!

4 c includes:

★ **A Source Code Analyzer:**
The analyzer maps and cross-references functions and variables to their respective path and file names. The cross-reference table can be displayed alphabetically and by type of variable. It allows a programmer to edit functions directly without specifying a file name.

★ **A Point-and-Shoot Zoom Key:**
Which instantly locates any function, global variable, structure definition, #define statement, or macro, etc., in any file, directory, and drive, and displays it in a new edit window.

★ **A Full Featured Built in Editor:**
Featuring multiple files, multiple windows, cut-and-paste between files, search-and-replace, auto indent, tab support, and a familiar user interface with pull-down menus, or use your own editor!

"Given a procedure's identifier, an editor should be able to bring up the full procedure interface, including parameters, so that you don't have to remember how many and in what order. Wouldn't it be nice to hit a hot key on the name of a complicated data type and have its definition pop up in a window? It may seem like a real small matter, but how many of you still flip through printouts doing things the editor should be doing?"

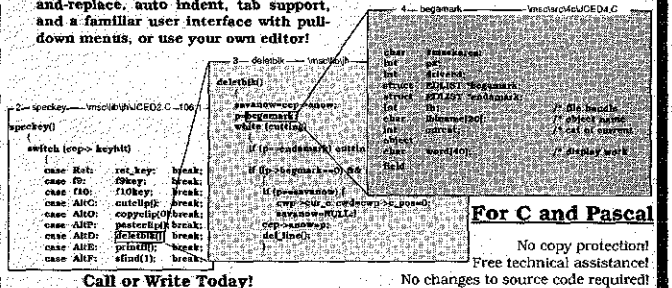
- Phillipe Kahn, "The Connection", Spring/Summer 1988

4 C does this and more

"4c's Magic is apparent..."
- PC Magazine, February 1989

"Daylight in the C Confusion"
- Toolbox magazine, December 1988

"...Destined to become the standard editor for C..."
- Computer Personlich magazine, October 1988



For C and Pascal

No copy protection!
Free technical assistance!
No changes to source code required!
Required IBM PC compatible with 350K RAM.

30 Day Money Back Guarantee!
Only $119.00 plus shipping
Visa, Mastercard, check, money order, or COD.

Call or Write Today!

Tri-Technology Systems Inc.
1225 S. Elgin, Forest Park, Il. 60130
Phone: 312-366-7595

Number **276** on the Reader Service Card

The DDE Message Set

The sidebar accompanying this article provides a fairly detailed description of the nine basic DDE messages in their main forms. Here we have tried to be a bit encyclopedic by pulling together information from a variety of sources into a kind of handbook. We hope it will be useful, especially to a programmer who is faced with the problem of implementing DDE in an application.

In the next installment, we will apply the information presented here to our communications server. PJ

William S. Hall is a senior engineer at CONNECT Inc. in Cupertino, California. He specializes in Windows communications software.

References

Hall, William S., "A Simple Debugging Utility for Presentation Manager and Windows," *Programmer's Journal* 7.2 (1989), 56–73.

Berger, Harvey, "Dynamic Data Exchange Enhances Application Connectivity," *Microsoft Systems Journal*, Vol. 1 No. 1 (1986), 7–15.

Welch, Kevin P., "Interprocess Communications Using Windows' Dynamic Data Exchange," *Microsoft Systems Journal*, Vol. 2 No. 5 (1987), 13–38.

HARVARD SOFTWORKS

NUMBER ONE IN FORTH INNOVATION

(513) 748-0390 P.O. Box 69, Springboro, OH 45066

MEET THAT DEADLINE ! ! !

- Combine the raw power of extensible languages with the convenience of carefully implemented functions!
- Execute faster than optimized C!
- Compile 40,000 lines per minute!
- Stay totally interactive, even while compiling!
- Program at any level of abstraction from machine code thru application specific language with equal ease and efficiency!
- Modify routines without recompiling!
- Use source code for 2500 functions!
- Use data structures, control structures, and interface protocols from any other language!
- Implement that borrowed feature, often more efficiently than in its source!
- Use an architecture that supports very small programs or full megabyte ones with a single version!
- Forget chaotic syntax requirements!
- Use subroutine libraries written for other languages! More efficiently!
- Outproduce experienced programmers stuck using conventional languages!

HS/FORTH with FOOPS - The only full multiple inheritance object oriented language under MSDOS!

WAKE UP ! ! !

Forth is no longer a language that tempts programmers with "great expectations", then frustrates them with the need to reinvent simple tools expected in any commercial language.

HS/FORTH Meets Your Needs!

We don't shortchange you with promises. We provide implemented functions to help you complete your application quickly.

Don't judge Forth by public domain products or ones from vendors primarily interested in consulting - they profit from not providing needed tools! Public domain versions are cheap - if your time is worthless. Useful only in learning Forth's basics, they fail at showing its true potential. Not to mention being s-l-o-w.

You can't add extensibility to fossilized compilers. You can easily add features from any other language to **HS/FORTH**. Try our full modifiable multiple inheritance **OOPs** extension or our **Fortran** translator & mathpak!

HS/FORTH has more features and libraries than we can possibly describe here. **Please ask for our brochure!**

HS/FORTH uses MSDOS or PCDOS. Each level includes features of lower ones. Level upgrades: $25. plus price difference. Sources in ordinary text files.

STUDENT LEVEL **$145.**
text & scaled/clipped graphic windows, ellipses, splines, arcs, fills, turtles; powerful parsing, formatting, file and device I/O; shells; interrupt handlers

PERSONAL LEVEL **$245.**
software floating point, trig, transcendental, quad & scaled integer, automatic optimizer for assembly like performance.

PROFESSIONAL LEVEL **$395.**
hardware floating point, turnkey, seal, Rosetta Stone Dynamic Linker, round robin & time slice multitaskers, dynamic string manager, file blocks, sector mapped blocks, assemblers

PRODUCTION LEVEL **$495.**
Metacompiler: DOS/ROM/direct/indirect C data structures & struct compiler, TurboWindow-C graphics library,

Options:
FOOPS+ with multiple inheritance **$ 75.**
286FORTH or **386FORTH** **$295.**
TurboWindow-C (Metagraphics) **$ 99.**
BTRIEVE for HS/FORTH (Novell) **$199.**
ROMULUS HS/FORTH from ROM **$ 95.**
FFORTRAN translator/mathpak **$ 75.**

Number **42** on the Reader Service Card

PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

C CODE FOR THE PC

source code, of course

	MS-DOS File Compatibility Package (read & write MS-DOS file systems on non-MS-DOS computers)	$750
NEW!	CSource Application Program Generator by MBAC (includes all source code; generator & libraries)	$500
	dB2c (dBase-to-C translator; includes db_Files for C and db_Tools for C)	$325
	CQL Query System (SQL retrievals on B-trees plus windows)	$325
	GraphiC 5.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
	Drasch Clisp with Crules (Lisp library and programming environment with rule processing capability; natural language example)	$300
	Greenleaf Data Windows (windows, menus, data entry, interactive form design; specify compiler)	$300
	PC Curses (Aspen, Software, System V compatible, extensive documentation)	$290
NEW!	C-Data Manager (object-oriented data management, persistent objects from runtime definitions, network and entity models)	$250
	MEWEL (extensible window and event library by Magma Software; message-passing & object-oriented; SAA-compatible; dialog editor)	$250
	TurboTEX (Release 2.0; HP, PS, dot drivers; CM fonts; LaTEX; MetaFont)	$250
	PC PostScript (complete PostScript interpreter (ROM Version 47.0A), 80286/386 only, many device drivers, optimized graphics, fast)	$250
	db_File & db_Retrieve (B-tree and network database with SQL query and report writer)	$245
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF; specify compiler)	$225
NEW!	Booter Toolkit (floppy disk bootstrap routines, DOS file system, light-weight multitasking, windows, fast memory management)	$210
	CDirect (multi-user hashed file manager; variable length fields, binary or ASCII data, alternate keys)	$210
NEW!	SilverComm (complete asynchronous communications library)	$210
	QuickGeometry Library (large collection of mathematics, graphics, display & DXF subroutines for CAD/CAM/CAE/CNC)	$170
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$165
	TurboGeometry (library of routines for computational geometry, Version 3.0)	$160
	AT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for ATs)	$160
	WKS Library Version 2.01 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	OS/88 (industrial-strength U**x-like operating system, many tools, cross-development from MS-DOS)	$150
	Cephes Mathematical Library (over 100 high-quality, double-precision scientific functions)	$150
	ME Version 2.1 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75)	$140
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
NEW!	Rogue Wave Vector & Matrix Classes (inc. C++ overloadings for standard operators, matrix inversion & FFT; Zortech or GNU C++)	$125
NEW!	Power Search by Blaise Computing (regular-expression compiler; generates machine code on the fly)	$120
	Install 2.3 (automatic installation program; user-selected partial installation; CRC checking)	$120
NEW!	Installation Toolkit 3.0 (250-page User's Guide, interactive script development program; many special cases covered)	$120
NEW!	B-Strings (dynamic string handling; cut, copy, paste, search, user input, etc.; non-fragmenting memory management)	$105
	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$105
	Minix Operating System (Version 1.3; U**x-like operating system, includes manual)	$105
	PC/IP (CMU/MIT TCP/IP for PCs; Ethernet, Appletalk & NETBIOS drivers, RVD, update by Dan Lanciani)	$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
	The Profiler (program execution profile tool)	$100
	QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
	Otter 1.0 (beautiful theorem-prover by Bill McCune; includes manual & two books by Wos; complete starter kit)	$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
	Make (macros, all languages, built-in rules)	$75
	eval() (C function to evaluate ASCII infix expression string; 17 built-in functions)	$75
	XT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for XTs)	$75
	Professional C Windows (lean & mean window and keyboard handler)	$70
	ScreenLib (simple screen definitions, windows, pop-up menus, context-sensitive help)	$65
	Heap Expander (virtual memory manager using expanded memory, extended memory, and disk space)	$65
	Quincy (interactive C interpreter)	$60
	Symtab/Ptree (general-purpose symbol table/parse tree construction and management package; specify Symtab or Ptree)	$60
	Coder's Prolog (Version 3.0; inference engine for use with C programs)	$60
	Async-Termio (Unix V compatible serial interface for MS-DOS; stty, ioctl, SIGINT, etc.)	$55
	Backup & Restore Utility by Blake McBride (multiple volumes, file compression & encryption)	$50
NEW!	Floppy TAR (TAR backup and restore on MS-DOS devices; direct access to non-standard devices)	$50
	SuperGrep (exceptionally fast, revolutionary text searching algorithm; also searches sub-directories)	$50
NEW!	REGX Plus (search and replace string manipulation routines based on regular expressions)	$50
	OBJASM (convert .obj files to .asm files; output is MASM compatible)	$50
	Multi-User BBS (chat, mail, menus, sysop displays; does not include Hayes modem driver)	$50
NEW!	LaplaceB (LaPlace polynomials, real and complex)	$50
Updated!	CLIPS (rule-based expert system generator, Version 4.3; advanced manuals available)	$50
NEW!	Kier DateLib (all kinds of date manipulation; translation, validation, formatting, & arithmetic)	$45
Updated!	Fortran-to-C Translator by Polyglot	$40
NEW!	DES Encryption & Decryption (2500 bits/second on 4.77 MHz PC for on-the-fly encryption at 2400 baud; U.S. only)	$40
NEW!	FlexList (doubly-linked lists of arbitrary data with multiple access methods)	$40
	Virtual Memory Manager by Blake McBride (LRU pager, dynamic swap file, image save/restore)	$40
	Heap I/O (treat all or part of a disk file as heap storage)	$40
Updated!	Bison & BYACC (YACC workalike parser generators; documentation; no restrictions on use of BYACC output)	$35
	PC-XINU (Comer's XINU operating system for PC)	$35
NEW!	RXC & EGREP (Regular Expression Compiler and Pattern Matching; RXC makes finite state machine from regular expression)	$35
NEW!	CCALC (handy extended-precision calculator; real and complex models; many built-in functions)	$30
NEW!	GNU Awk & Diff for PC (both programs in one package)	$30
	6-Pack of Editors (six public domain editors for use, study & hacking)	$30
	Crunch Pack (14 file compression & expansion programs)	$30
	Pascal P-Code Compiler & Interpreter or Pascal-to-C Translator (Wirth standard Pascal)	$25
	FLEX (fast lexical analyzer generator; new, improved LEX; official BSD Version 2.1 with docs)	$25
	List-Pac (C functions for lists, stacks, and queues)	$25
	A68 (68000 cross-assembler)	$20
	XLT Macro Processor (general purpose text translator)	$20
	Data	
NEW!	Moby Pronunciator (150,000 words & phrases encoded with full IPA pronunciation & emphasis points; 900 distinguished by part-of-speech)	$160
NEW!	Moby Part-of-Speech (200,000 words and phrases described by prioritized part(s)-of-speech)	$105
NEW!	Moby Hyphenator (150,000 words fully hyphenated/syllabified)	$120
	Smithsonian Astronomical Observatory Subset (right ascension, declination, & magnitude of 258,997 stars)	$60
	Moby Words (500,000 words & phrases, 9,000 stars, 15,000 names)	$65
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
	KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
	USNO Interactive Computer Ephemeris (high-precision moon, sun, planet & star positions)	$30
	U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA

acw!info@uunet.uu.net

Voice: (512) 258-0785
BBS: (512) 258-8891
FAX: (512) 258-1342

Free surface shipping for cash in advance **For delivery in Texas add 7%** **MasterCard/VISA**

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Using Dynamic Memory Efficiently

by

John Otken

In previous columns, I developed an efficient methodology for assembly language programming. This column applies the information to a program, "DB," which is a variation on the hex dump program everyone writes at some point in his or her life.

Instead of the familiar hex dump, DB formats its output into assembly language "data byte" statements common to most assemblers, 80X86 in particular. When I want to encode file contents into a program, I process the file with DB and include the output in the source. I use this in stand-alone programs that download code to embedded systems.

DB source code appears in listings 1 and 2. The second listing contains miscellaneous library routines, most of which appeared in earlier columns. Extract the new routines from this source and include them in your library. If you cannot decide how to split the file into modules, get hints from the public statements.

The only company that can add structure to 80X86 assembly language is Microsoft, but they have never shown any inspiration with MASM.

When you examine Listing 2, you may frown on my use of an **ms_dos** procedure as wasteful: Why call a two-byte instruction? Besides the debugging possibilities, you could have an easy way to convert your programs to OS/2 (or UNIX, OS/3, etc.) when it becomes popular.

In any case, an INT 21 simulator shouldn't take more than a day to write. You have to handle the command line and environment, but they pose no significant problems. "Porting" most programs to the new operating system requires only a link operation.

I don't have OS/2 on any of my systems, but I intend to clean up my spare 386 for either OS/2 or UNIX. If I install OS/2, an INT 21 simulator sounds like a great first project, and you can read about it in a future column.

The present column concerns a DOS program for which the primary code appears in Listing 1. Notice that the program contains no segment overrides and modifies segment registers in only a few places. The use of "BP indexing" implies a stack segment override, thus providing access to DGROUP static variables regardless of the contents of registers DS or ES. If you have ever fought with segment registers in an assembly language program, this technique is very valuable (for more details, see *PJ* 7.6).

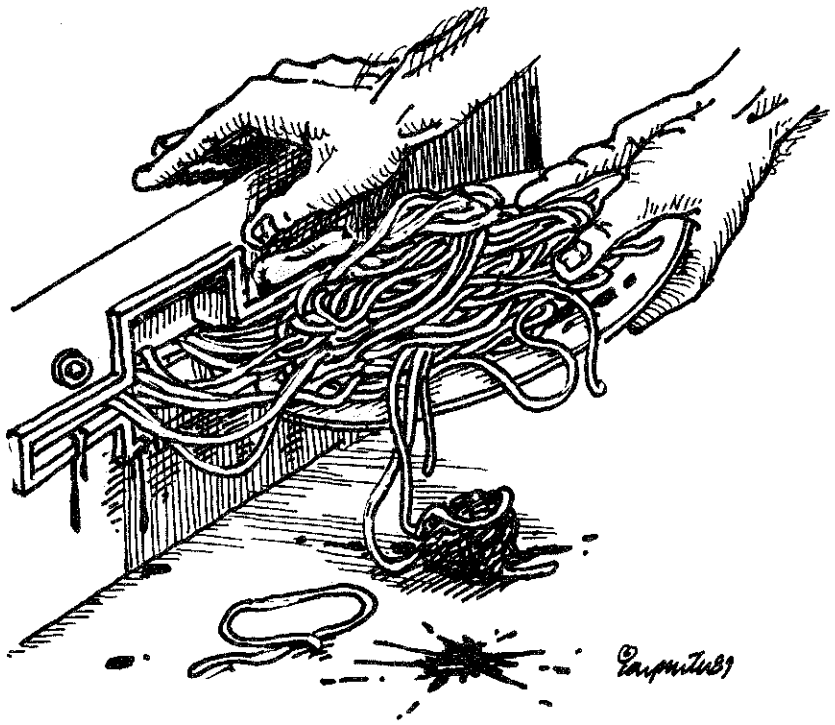
While BP indexing increases performance, its primary purpose is to simplify programming. I continue to avoid the topic of optimization because the popular discussions of optimization offer negligible enhancements. According to Intel's published instruction timing, the 486 actually executes "tightly" optimized assembly language slower!

This program is the last to use an explicit "preamble" consisting of lines from the **dosseg** directive through the equates. This section of statements is common to all programs, and future examples will use a standard include file.

Besides the segment declarations and universal equates, the include file contains useful macro definitions, many of which define convenient OPTASM features missing from TASM and MASM. I'll introduce the include file in the next column.

Using this program as a software test bed permits me to preview the include file contents and several potential library routines. I have an opportunity to improve their design without affecting other programs. Remember, good design results from experimentation.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Software Test Beds

Software test beds are an effective way to verify program elements, which is important for library routines and include files. Debugging is only one of several steps during test-bed development. You should experiment with different entry and exit conditions for procedures. Check aesthetics too, because "clean" code is easier to program.

If the routines are critical and you know the development process is weeks or months long, give the routines a special name that indicates their experimental status. The Air Forces uses the prefix "X-", which is good enough for me.

With designated experimental names, you can add the routines to your library and force dependent programs to change when you change the names in the library. You should change a procedure name whenever you alter code in a way that could affect dependent programs (bug fixes excepted).

For example, my hash routines went through several phases of development. The first version was a test for the interface; the routines processed a nonhashed symbol table. I began using the routines in programs and found an acceptable interface. Because the routines had performance problems, I rewrote them with true hashing algorithms.

The DB program has several elements I want to scrutinize before moving them to a library or include file: I want to check the aesthetics of the equates, test the error handler, experiment with the dynamic storage and file I/O interface, and play with the command-line scanner.

Set_argc_argv operates like the C command-line scanner. I tried several scanner interfaces before I settled on the "far pointer to an array of offsets" approach. The scanner is functional but needs work on "quoted" arguments and wildcards.

I also borrow the notion of "NULL" from C. Using NULL instead of 0 indicates the expression involves a pointer. From the following C fragment, a programmer knows that *p* is a pointer:

```
if (p == NULL)   ...
```

C permits a program to associate types with variable names. In 80X86 assembly language, the primary "variables" are the register names. Any clue to the type of data in a register improves readability.

I have a collection of NULL symbols defined for different types of data. "NULL_POINTER" is similar to NULL in C. (I did not use NULL in assembly language because Microsoft defines a segment named NULL.) In the following fragment, the comparison of SI and NULL_POINTER tells the programmer that the value in SI is a pointer:

```
cmp     si,NULL_POINTER
jne     fehl            ; if error text found
```

"NULL_CHAR" is a character. I also define "_CHAR" equates for nondisplayable characters: "CR_CHAR" is carriage return; "SPACE_CHAR" is a blank:

```
cmp     al,NULL_CHAR
je      dazl            ; if end of text
```

The last equate from DB is "wptr". I equate this symbol to the text string "word ptr". This is a convenient abbreviation I use to keep lines as short as possible. I begin comments in column forty, and I try to avoid instruction running into the comment area. Similar equates exist for "byte ptr" and "dword ptr".

DB belongs to a popular class of programs that read from a single file and write to the console. I use the acronym FIDO, meaning "file in display out."

I use the FIDO model for most utilities, even when direct file output would seem more appropriate. Console output is simple to use, avoids complex buffering, and supports redirection to a file. Development is safe and easy because debuggers readily display program output.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

In my column on assembly language libraries (*PJ* 7.5), I provided a "template" for library modules. One can easily rip out the procedures specific to DB and turn this source into a template for FIDO programs. (Templates are great time savers; I'll spend more time on them in future columns.)

Start-Up Procedure

Another notion borrowed from high-level languages such as C is generic start-up code. Once you standardize on a memory model, most programs initialize their environment in the same way. In DB, the start-up procedure, **start_up**, performs the following functions:

- Clears the direction flag in the processor status word. This ensures that string instructions advance through memory with increasing address values.
- Sets the stack frame register BP. In contrast to high-level-language programs, the stack frame rarely changes in my programs.
- Saves the program segment prefix, or PSP, which is the base address for a program. You need the PSP to find the environment area and command line. There are two DOS functions that return the PSP. One function requires version 3 of DOS, and Microsoft does not document the other function. **Start_up** saves the PSP from register ES, which Microsoft documents as containing the PSP when a program begins execution.

  If this program defined locations in segment BSS (with .data?), the start-up code should zero the segment. This program does not use BSS and does not have code to zero it. I must add this function before moving **start_up** to a library.
- Initializes dynamic storage by shrinking the program's memory bucket to a minimum size. In the small model, the **dosseg** directive forces the linker to position the stack last; thus, the initial stack pointer is also the last position in the program. Normalize this address and subtract the PSP to compute the program size in paragraphs. DOS function 4Ah shrinks the program's memory block.

  Segment arithmetic doesn't sit well with OS/2, but one purpose of the start-up code is to isolate potential incompatibilities. As with the **ms_dos** procedure, **start_up** must change for OS/2.
- Scans the command line, counts the tokens and builds a list of offsets of asciiz strings.

Command-Line Scanning

This program mimics the C mechanism for passing command-line parameters to a program. The **set_argc_argv** procedure separates the command line into asciiz tokens and builds an array of offsets for each token. A pointer to the array is in static variable *argv*, and the total number of tokens including the program name is in *argc*.

Since DB is a test-bed program, **set_argc_argv** is one of the routines still under development. As I mentioned earlier, it needs support for quoted tokens and wildcards.

Set_argc_argv has 10 labels and 11 jumps or loops. Given so many jumps, this routine might qualify as spaghetti code (or, as we say in Los Angeles, "pasta" code).

"Coding Up Pasta"

I have a simple test that shows what kind of noodles you "cook up" in a procedure. Draw a line from jumps and loops to their targets and count the number of intersections. Do not count conditional jumps for abnormal or error conditions (lest you ignore errors to improve your score). Look up the number of intersections in the following table:

0-1	linear code, "pasta-grata"
2-4	fettuccine
5-8	linguine
9+	angel hair, "pasta-nongrata"

This test does not penalize code using traditional block-structure constructions such as while loops or simple conditionals. It does penalize "if then else"—as it should, because such code sequences are unfortunately complicated (score one for high-level languages).

I wish I could offer a perfect solution to this problem, but 80X86 assembly language is not block structured. "Grafting" block structure to your programs with macros and preprocessors can easily ruin it. The only company that can add structure to 80X86 assembly language is Microsoft, but they have never shown any inspiration with MASM. What would you expect when you write an assembler in C?

While we cannot avoid complex code sequences, we can minimize their damage. I covered several techniques in my column in *PJ* 7.4; here's a brief summary:

- Organize and comment your code into functional blocks.
- Comment conditional jumps.
- Use local labels within a procedure.
- Spend more time commenting complex code (quality not quantity).

Intersecting jumps are easy to eliminate from some code sequences. Consider the sample below, which sets register AX to 1 if it contains the character "Y"; otherwise, it sets AX to zero:

```
        cmp     al,'Y'
        je      1$              ; if yes
        mov     ax,0            ; else no
        jmp     2$
1$:     mov     ax,1
2$:     ...
```

The following, better sequence compares AL and then loads a default into AX before the conditional jump. This is somewhat tricky, but it's easy to understand after the first time. Avoid placing too many instructions between the compare and the conditional jump.

```
        cmp     al,'Y'
        mov     ax,1
        je      1$              ; if yes
        mov     ax,0            ; else no
1$:     ...
```

In contrast to the spaghetti in **set_argc_argv**, **read_entire_file** is an excellent example of picture-perfect linear code. This procedure has no jumps or loops at all (error jumps excepted); the CPU follows a single

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Would you like to be able to include a hypertext help system in your programs like the big guys: **Microsoft and Borland?***

Introducing... Help System II

We've enhanced the hypertext help system our hypertext products have used for 2 years so that you can include them in your programs:

- **Context Sensitivity**
- **Help for 84 different key combinations**
- **Requires less than 10K**
- **Data compression up to 80 percent**

$199.95
buys object files for Pascal, C, Basic and technical support

To order: **(203) 368 - 0632**
NTERGAID Inc.
955 Connecticut Ave.
Bridgeport, CT 06607

*** At least one of these companies purchased our product before they released theirs.**

Number **302** on the Reader Service Card

The Heap Expander™ version 2.05

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

only $79.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:

- Turbo C
- Microsoft C 4.0 and 5.0
- Turbo Pascal 3.0 and 4.0

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*CA residents add 6 1/2% sales tax. Foreign customers, please add $4.00 for shipping and handling.

The Tool Makers
P. O. Box 2151
Santa Cruz, CA 95063
(408) 458-0690

Number **166** on the Reader Service Card

Performance Programming TOOLS

For IBM PC, XT, AT, PS/2 & compatibles

Commenting Disassembler

Sourcer™ creates detailed commented assembly source code and listings from memory and executable files. Built in data analyzer and code simulator produce great results. Supports 8088/87 through 80386/80387 and 80486 instruction sets. *"Sourcer is the best disassembler we've ever seen!" PC Magazine, January 17, 1989, page 101.* $99.95

BIOS Source

The **BIOS Pre-Processor™** with Sourcer provides the first means to obtain accurate legal source listings for any BIOS. Identifies entry points with detailed in-line comments, explains functions, and more! Requires Sourcer. $49.95; Sourcer & BIOS $139.95 (Save $10)

Unpack EXE Files

The **Unpacker™** group of programs assists with the debugging and analysis of software. Unpack packed EXE files and convert multi-segment COM files back to EXE. Includes handy interrupt viewer and video test program. $39.95.

Assembly Code Analyzer

ASMtool™ is an assembly language source code analysis tool that simplifies the understanding of complex software. Performs automatic flowcharting, analysis of stack and register usage, procedure tree diagrams, and static timing analysis. $89.95

Zoomable File Viewer

With **View-It™** zoom out to see up to 8 times more information on a single screen. View-It handles files of any size, provides dual windows, advanced search and mouse support, and is incredibly fast! EGA or VGA required for zoom feature. $69.95

Assembly Tools Package

ASM ProPak™ includes Sourcer with BIOS Pre-Processor, Unpacker, ASMtool and View-It. $299.75 (Save $50)

C Code Your Way

C-Clearly™ formats C source code according to your style preferences. Make your own style template or use and modify supplied templates such as standard K & R and others. Also produces formatted listings with optional flow lines, line numbers, headers/footers and more. $129.95

Shipping & Handling: USA: $3; Canada/Mexico $10; Other Countries $15; CA Residents add sales tax. Visa/MasterCard/COD accepted. If within 30 days of purchase you find our product does not perform in accordance with our claims, call our customer service department and we will gladly arrange a refund.

NOT COPY PROTECTED
30-DAY MONEY BACK GUARANTEE

For orders or information call:

1-800-662-8266

V COMMUNICATIONS, INC.
3031 Tisch Way, Suite 802, Dept. PJ5
San Jose, CA 95128 (408) 296-4224

Number **125** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

straight down path through the code, and the logic is easy to follow.

File I/O

There are many different ways to read a file, but only two methods are easy. The first method is the slow but sure "get one character" (**fgetc**), common in C libraries. The second, faster method is to read the entire file into a single large buffer.

I implement the later method in DB because it permits an incredibly simple interface: Call **read_entire_file** with a filename, and it returns a pointer to the entire file contents in memory. At this point, the file is easy to manipulate. For example, the following instructions send the file contents to the console:

```
mov     ah,40h ; write to standard output
mov     bx,1
mov     dx,si
call    ms_dos
```

The only limitation of **read_entire_file** is that the file size cannot exceed available memory or 64K, whichever is less. This practically eliminates its use in commercial applications and even certain utilities, but its ease of use outweighs this limitation.

Read_entire_file operates with either binary or ASCII data. The routine places a zero byte after the end of file, which simplifies work with ASCII files. Other file I/O routines are quite complex by comparison. My personal library has separate modules for binary and ASCII data, with 1000 source lines exist between them; they perform the same function that **read_entire_file** does with less than 100.

The division of labor between the binary and ASCII routines is interesting. The binary routines call the operating system and conduct simple buffer management. The ASCII routines call the binary routines for their functions and then perform additional "ASCII" processing. In effect, the ASCII routines form a "layer" on top of the binary routines.

Layering

The goal of layering is reduced complexity. Each layer has a specific function and operates independently except for communications to an adjacent layer through an interface.

The collection of an application program, MS-DOS, and the BIOS form a well-known example of layering. The top layer is the application that knows how to process a file but relies on MS-DOS to find and read the file. MS-DOS occupies the middle layer; it knows the structure of directories on the disk but relies on the BIOS to transfer data from the disk. The bottom layer is the BIOS, which knows how to read the disk but does not know what the data mean. Because the BIOS isolates the hardware from MS-DOS and MS-DOS isolates the file structure from the application, a program can execute on thousands of different hardware platforms.

Programming a layer is straightforward: Start by assigning ownership of an object or resource to a module; then build an interface to the object or resource through routines in the module. You now have a layer.

Of course, a layer does not have to exist as a separate module. You can create a layer with a single procedure. In DB, **read_command_line** is a layer because no part of the program can access the command line without calling **read_command_line**. Likewise, **read_environment** owns the environment. The idea of ownership is by convention only; there are no checks to enforce it.

Layered design is a form of modularization. One can compare layers with procedures by analogy. Procedures are collections of instructions, and layers are collections of procedures. You should not reach into the middle of a procedure, and you should not reach into the middle of a layer. Both benefit from a well-defined interface.

You should arrange layers in a strict hierarchy. If a high level accesses a low level—bypassing a middle level—you lose the benefits of layering. Common examples include DOS applications that access the BIOS directly. Strictly speaking, DOS owns the BIOS. Applications that call the BIOS sacrifice portability and, as anyone who tries to run such a program on a Japanese PC quickly discovers:

- Layered code is easier to debug.
- Layered code is modular; so, layers can be replaced.
- Layered projects have natural divisions for group efforts.

You can apply layering to all kinds of projects. Graphic interfaces are a good example. The X Window System, also known as X, provides a simple window manager over which one might layer a system for pull-down windows and dialog boxes.

Another example is dynamic storage. The DOS interface to dynamic storage is so bad, I hide it with a layer. The routines **malloc_raw** and **free_raw** provide an acceptable interface to dynamic storage.

Dynamic Storage

When you need a large section of memory in a program, you can allocate storage directly in the program data space; this is static allocation. Alternatively, your program can call the operating system to allocate memory; this is dynamic allocation. The latter choice offers more flexibility, because you may not know how much memory you need until execution.

```
        .data?
static_storage db 512 dup(?) ; allocate static memory
```

Such is the case with **read_entire_file**, which allocates a file buffer from dynamic storage. Before allocation, **read_entire_file** looks up the file size and then requests that many bytes of memory.

Set_argc_argv allocates a small piece of storage for the command-line parameters. (I did not use dgroup space because I anticipate support for wildcards in future versions. Wildcard arguments in a large directory can generate a sizable parameter list.) The first word from the procedure names **free_raw** and **malloc_raw** derive from their C counterparts. I use the suffix "raw" to show these routines occupy a low layer in my scheme for storage management. **Free_raw** and **malloc_raw** are now the highest-level interface to storage, but I plan to overlay them with more layers.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programming Glossary

You can use language to enhance program readability in surprising ways. The use of the "raw" suffix is one example. You should keep a glossary with notes and definitions on your personal programming "dialect." Group programming projects should find glossaries particularly important.

Other entries from my personal glossary of "procedure suffixes" include "read", which means the procedure returns a pointer in DS:SI; "write", which indicates a returned pointer in ES:DI; and "ok", which means errors have special handling.

Prefixes are often effective ways to identify a procedure's major function at a glance. For example, I prefix all hash-management routines with "hash". Can you guess the name of my library function that returns a pointer to a hash table entry in DS:SI?

Your glossary also can apply to comments. "If" begins comments on conditional jumps instructions. I "advance" a pointer (or counter) by adding or subtracting:

```
mov     ax,symtab_count ; advance symbol count
inc     ax
jz      6$              ;  if too many symbols
mov     symtab_count,ax
```

I have specific meanings for terms relating to dynamic storage. *Bucket* is a fixed section of physical storage. **Malloc_raw** and **free_raw** manage storage buckets. *Buffers* are storage buckets that can move or change size. *Blocks* are like buffers, but with a fixed length.

Conclusion

Be sure to call my bulletin board system with any questions or comments you have about this column; the number is 213-559-1449. My thanks to those who have already called. If you write a program using the information in this column, I would like to see a short description of what the program does and any problems you had writing it.

I collect excerpts from the column and all sample programs into a file you can download. File relating to this column are pj81.arc, pj76.arc, pj75.arc, and pj74.arc. The digits represent the particular issue number. These files are also available in "zip" format.

Next time, I'll finish the start-up code and the standard include file. PJ

John Otken is the president of Soft Advances, where he develops software for embedded systems when he isn't snow-skiing. When the powder is deep at Mammoth, good luck reaching him at 10811 Washington Blvd., Suite 205, Culver City, CA 90232, 213-559-7015.

Code follows.

PJ CODE
All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 88.

Code from "Using Dynamic Memory . . . ," by John Otken — 1

```
LISTING 1

        title   dump file as db statements
        dosseg
        .model  small
NULL    segment para public 'BEGDATA'
NULL    ends

NULL_POINTER    equ     0
NULL_CHAR       equ     0
CR_CHAR         equ     13
SPACE_CHAR      equ     32
wptr            equ     word ptr

        .stack
        .data
dgroup_start    label   word

psp             dw      0
argc            dw      0
argv            dd      0

error_text      dw      0       ; ascii: error msg offset in DGROUP

db_txt          db      ',      db      ',0

err_default     db      'Error',0
err_no_memory   db      'Insufficient memory',0
err_too_big     db      'malloc exceeds maximum',0

err_file_not_found  db  'File not found',0
err_file_io         db  'File I/O error',0
err_file_too_big    db  'File too big',0

help_text  db  'db dumps file as assembly language db statements',13,10,0

        .code
        extrn exit_program:near,exit_with_error:near,ms_dos:near,outchr:near
        extrn outchr_ascii:near,dgroup:near,outchr_hex_byte:near,outchr_newline:near
        extrn strcpy:near,strskp:near,strskp_white:near

;;      fatal error handler
;;
fatal_error_handler proc
        mov     ax,NULL_POINTER         ; display error message
        xchg    ax,error_text[bp]
        cmp     ax,NULL_POINTER
        jne     fehl                    ;  if error text found
        lea     ax,err_default[bp]      ;  else use default error text
fehl:   call    outchr_ascii,dgroup

        jmp     exit_with_error
fatal_error_handler endp

;;
;;      free raw
;;
;;      entry   ES:DI   storage bucket ptr
;;      exit    CF      if bad ptr
;;      uses    AX
;;
free_raw proc
        mov     ah,49h
        jmp     ms_dos
free_raw endp
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Otken code, continued

2

```
;;      malloc raw
;
;       entry   CX      request bytes size (0<x<0FFF0h)
;       exit    ES:DI   storage bucket pointer
;               Cf      if not enough memory or request too large
;       uses    AX
;
malloc_raw proc
        push    bx                      ; convert byte count to
        mov     bx,cx                   ;  paragraph count
        add     bx,15
        jc      mar2                    ;  if too much requested
        shr     bx,1
        shr     bx,1
        shr     bx,1
        shr     bx,1

        mov     ah,48h                  ; request memory from dos
        call    ms_dos
        jc      mar3                    ;  if not enough memory
        mov     es,ax
        mov     di,0

mar1:   pop     bx
        ret

mar2:   mov     error_text[bp],offset err_too_big
        jmp     mar1
mar3:   mov     error_text[bp],offset err_no_memory
        jmp     mar1
malloc_raw endp


;;      main
;
main    proc
        mov     ax,@data                ; set dgroup stack
        mov     ss,ax
        mov     sp,offset stack
        call    start_up

        cmp     argc[bp],1
        jbe     mai1                    ;  if no parameters - display help

        lds     si,argv[bp]             ; get 1st command line parameter
        mov     si,[si+2]

        call    read_entire_file
        jc      mai2                    ;  if error reading file
        call    outchr_db_bytes         ; convert file to db statements
        jmp     exit_program

mai1:   lea     ax,help_text[bp]        ; display help text
        call    outchr_asciiz_dgroup
        jmp     exit_program

mai2:   jmp     fatal_error_handler
main    endp


;;      open input file
;
;       entry   DS:SI   asciiz file path
;       exit    AX,BX   file handle
;               Cf      if error
;
open_input_file proc
        mov     ax,3D00h
        xchg    dx,si
        call    ms_dos
        xchg    dx,si
        mov     bx,ax
        ret
open_input_file endp
```

3

```
;;      outchr memory byte
;
;       entry   DS:SI   byte pointer
;       exit    SI      updated
;       uses    AX
;
outchr_memory_byte proc
        mov     al,'0'
        call    outchr
        lodsb
        call    outchr_hex_byte
        mov     al,'h'
        jmp     outchr
outchr_memory_byte endp


;;      outchr db bytes
;
;       entry   DS:SI   memory pointer
;               CX      byte count
;       uses    AX,CX,DX,SI
;
outchr_db_bytes proc
        jcxz    odb4                    ;  if no bytes

odb1:   lea     ax,db_txt[bp]           ; *     db      *
        call    outchr_asciiz_dgroup

        mov     dx,8                    ; set count for bytes on this line
        cmp     cx,dx                   ;  minimum of 8 or bytes left
        ja      odb2
        mov     dx,cx

odb2:   call    outchr_memory_byte      ; display memory bytes
        dec     dx
        jz      odb3                    ;  if 16th byte on current line
        mov     al,','
        call    outchr
        loop    odb2                    ;  if more bytes
        inc     cx                      ;  (adjust CX for loop below)
odb3:   call    outchr_newline
        loop    odb1                    ;  if more bytes

odb4:   ret
outchr_db_bytes endp


;;      read command line
;
;       exit    DS:SI   program command line
;
read_command_line proc
        mov     ds,psp[bp]
        mov     si,81h
        ret
read_command_line endp


;;      read entire file
;
;       entry   DS:SI   asciiz file name
;       exit    DS:SI   ptr to storage containing file (NULL delimited)
;               CX      file byte count
;               Cf      if file not found or too big
;       uses    AX
;
read_entire_file proc
        push    bx
        push    dx
        push    di
        push    es
        call    open_input_file
        jc      ref3                    ; if file not found
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

4

```
        mov     ax,4202h                ; get file size
        mov     cx,0
        mov     dx,0
        call    ms_dos
        jc      ref4                    ;  if unexpected file i/o error

        add     ax,1                    ; bump file size for delimiting zero
        adc     dx,0
        jnz     ref5                    ;  if file too big (>64k)

        mov     cx,ax                   ;  allocate storage for file contents
        call    malloc_raw
        jc      ref1                    ;  if insufficient memory

        mov     ax,4200h                ; rewind file
        mov     cx,0
        mov     dx,0
        call    ms_dos
        jc      ref4                    ;  if unexpected file i/o error

        mov     ah,3Fh                  ; read entire file
        mov     cx,-1
        mov     dx,di
        push    es
        pop     ds
        call    ms_dos
        jc      ref4                    ;  if unexpected file i/o error

        mov     cx,ax                   ; return file byte count
        mov     si,di                   ;  and storage pointer

        add     di,ax                   ; delimit file with NULL
        mov     al,NULL_CHAR
        stosb

ref1:   mov     ah,3Eh                  ; close file
        pushf                           ;  (preserve carry flag)
        call    ms_dos
        popf
ref2:   pop     es
        pop     di
        pop     dx
        pop     bx
        ret

ref3:   mov     error_text[bp],offset err_file_not_found
        jmp     ref2

ref4:   mov     error_text[bp],offset err_file_io
        jmp     ref1

ref5:   mov     error_text[bp],offset err_file_too_big
        stc
        jmp     ref1
read_entire_file endp


;;      read environment
;
;       exit    DS:SI   environment
;
read_environment proc
        mov     si,0
        mov     ds,psp[bp]
        mov     ds,[si+2Ch]
        ret
read_environment endp


;;      set argc argv
;
;       exit    Cf      if no memory
;       uses    AX,CX,DI,SI,ES,DS
;
```

5

```
set_argc_argv proc
        mov     cx,256                  ; allocate storage for parameters
        call    malloc_raw
        jc      saa9                    ;  if no memory

        mov     argc[bp],1
        mov     wptr argv[bp],di
        mov     wptr argv[bp+2],es

        mov     ah,30h                  ; check DOS version
        call    ms_dos
        cmp     al,3
        jb      saa10                   ;  if prior to DOS 3.x

        call    read_environment        ; skip environment (program name
saa1:   call    strskp                  ;  is found after environment in
        lodsb                           ;  DOS 3.x)
        cmp     al,NULL_CHAR
        jne     saa1
        lodsw                           ;  (skip 1)

        call    strcpy                  ; use program name as argv[0]
        inc     di

saa2:   call    read_command_line
saa3:   call    strskp_white            ; extract next command line parameter
        cmp     al,CR_CHAR
        je      saa7                    ;  if not end of command line

        inc     argc[bp]                ; advance parameter count
        jmp     saa5                    ; copy paramster
saa4:   stosb
saa5:   lodsb
        cmp     al,SPACE_CHAR
        jbe     saa6                    ;  if delimiter
        cmp     al,','
        je      saa6                    ;  if separator
        cmp     al,'~'
        jbe     saa4                    ;  if valid character (not delimiter)

saa6:   cmp     al,CR_CHAR              ; delimit parameter
        mov     al,NULL_CHAR
        stosb
        jne     saa3                    ;  if not end of command line

saa7:   inc     di                      ; word align array of offsets
        and     di,-2

        lds     si,argv[bp]             ; get ptr to first parameter
        mov     wptr argv[bp],di        ; set pointer list offset
        mov     cx,argc[bp]             ; get list count (cannot==0)

saa8:   mov     ax,si                   ; build array of offsets to each
        stosw                           ;  parameter
        call    strskp
        loop    saa8
        clc
saa9:   ret

saa10:  mov     ax,'C'                  ; microsoft c uses argv[0]=="C"
        stosw                           ;  for dos 2
        jmp     saa2
set_argc_argv endp


;;      start up
;
;       entry   ES      program segment prefix
;               SS      dgroup
;       exit    BP      dgroup base offset (always 0)
;       may use AX,BX,CX,DX,DI,SI,DS,ES
;
start_up proc
        cld
        mov     bp,offset dgroup_start
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Otken code, continued

6

```
        mov     psp[bp],es

        mov     ax,offset stack+15      ; compute number of paragraphs
        mov     cl,4                    ;  in program
        shr     ax,cl
        mov     bx,ss
        add     bx,ax
        sub     bx,psp[bp]

        mov     ah,4Ah                  ; shrink program memory block for
        call    ms_dos                  ;  dynamic storage system

        call    set_argc_argv
        ret
start_up endp

        end     main
```

LISTING 2

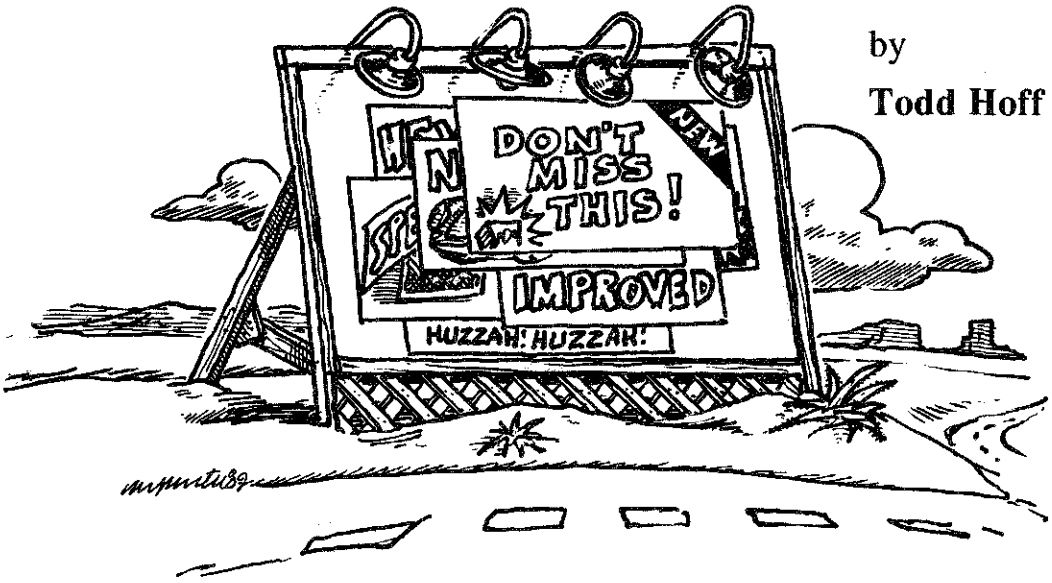
```
; this module contains new miscellaneous library routines for the db
; program.  other library routines are from previous columns.  you should
; split this file into modules for your assembly language library.

NULL_CHAR       equ     0
TAB_CHAR        equ     9
LF_CHAR         equ     10
CR_CHAR         equ     13
SPACE_CHAR      equ     32

        .model small
        .code
 public exit_with_error
 public outchr_asciiz,outchr_asciiz_dgroup
 public outchr_hex_word,outchr_hex_byte
 public outchr_newline
 public ms_dos
 public strcpy
 public strskp_white


;;      exit with error
;
exit_with_error proc
        mov     ax,4C01h
        jmp     ms_dos
exit_with_error endp


;;      ms dos
;
ms_dos  proc
        int     21h
        ret
ms_dos  endp


;;      outchr asciiz
;
;       entry   DS:SI   asciiz string
;       exit    SI      updated past NULL
;       uses    AX
;
outchr_asciiz proc
        lodsb
        cmp     al,NULL_CHAR
        je      oaz1
        call    outchr
        jmp     outchr_asciiz
oaz1:   ret
outchr_asciiz endp
```

7

```
;;      outchr hex word
;
;       entry   AX      word
;       uses    AX
;
outchr_hex_word proc
        push    ax
        mov     al,ah
        call    outchr_hex_byte
        pop     ax
        jmp     outchr_hex_byte
outchr_hex_word endp


;;      outchr hex byte
;
;       entry   AL      byte
;       uses    AX
;
outchr_hex_byte proc
        push    ax
        sar     al,1
        sar     al,1
        sar     al,1
        sar     al,1
        call    ohb1
        pop     ax
ohb1:   and     al,0Fh
        add     al,90h
        daa
        adc     al,40h
        daa
        jmp     outchr
outchr_hex_byte endp


;;      strcpy
;
;       entry   DS:SI   source ptr
;               ES:DI   destination ptr
;       exit    SI      updated past NULL
;               DI      updated, points to NULL
;       uses    AX
;
strcpy  proc
        lodsb
        stosb
        cmp     al,NULL_CHAR
        jne     strcpy
        dec     di
        ret
strcpy  endp


;;      strskp white
;
;       entry   DS:SI   text ptr
;       exit    SI      updated past spaces and tabs
;               AL      non-white character
;
strskp_white proc
        lodsb
        cmp     al,SPACE_CHAR
        je      strskp_white
        cmp     al,TAB_CHAR
        je      strskp_white
        dec     si
        ret
strskp_white endp

        end
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Using the DESQview API



by
Todd Hoff

Programming under MS-DOS can be a huge pain in the disk. Often, the creative art of programming is replaced by the tedium of compatibility issues and the frustration of trying to add program features far beyond the capabilities of DOS. Writing any substantial application often requires multiple libraries to accomplish anything, and advanced features like multitasking appear to be an MS-DOS dream.

Moving to OS/2, Windows, or even UNIX can solve many of these problems. Each of these choices, however, poses a serious problem for developers. Literally millions of XT- and AT-class machines will not be upgraded to run OS/2; developers abandoning this market will lose customers and money. Windows is still slow, and UNIX has its own set of problems.

DESQview and DESQview's Application Programming Interface (API) offers an attractive alternative, one that maintains compatibility with current systems while providing many advanced capabilities.

Most programmers have heard about Quarterdeck's DESQview program, but many may not be aware of the more than 200 C language library functions that give full access to DESQview's API. Using the API, developers can easily write programs that take advantage of DESQview's windowing, multitasking, menu-building, mousing, and context-sensitive help capabilities. Quarterdeck also provides a whole developer's toolbox centered around DESQview, including a debugger, a screen design tool, and an expanded-memory manager. This article covers the DESQview API C library.

What is DESQview?

DESQview is a multitasking operating system written atop the vanilla operating system functions offered by MS-DOS. DESQview transforms MS-DOS into an operating system with features approaching those of UNIX and OS/2, without sacrificing hardware or software compatibility. In fact, allowing current software to run unchanged in DESQview's multitasking windowing environment on all IBM PC platforms is DESQview's primary goal: Multiple programs, such as Ventura Publisher, Lotus 1-2-3, and dBASE, run simultaneously, without change, on 8086, 8088, 80286, and 80386 IBM PCs, IBM PS/2s, and compatibles.

From a top-down perspective, DESQview implements an extended or virtual machine. Above the hardware, above DOS, DESQview provides a layer of abstraction that hides operating system complexity through a simple interface. From the bottom-up view, DESQview is a resource manager allowing for controlled orderly access to CPU time, memory, and devices among competing tasks. The API serves as a set of system calls providing access to these operating system services.

On to the API

The API has two interfaces: an assembly language interface and a C library interface. Both can be used targeting different memory models and C compilers. Quarterdeck recommends using the large-memory model to avoid explicitly using FAR pointers.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

api_	General functions don't all operate on objects but cover miscellaneous capabilities and housekeeping.
app_	Application-management functions operate at the application level through application handles.
fld_	Field-management functions allow for defining and using fields within a window.
key_	Keyboard-management functions operate on keyboard objects.
mal_	Mailbox-management functions operate on mailbox objects.
obq_	Objectq-management functions operate on objectq objects.
pan_	Panel-management functions handle screen designs for menus, help screens, dialog boxes, and the like.
ptr_	Pointer-management functions operate on pointer objects.
qry_	Query-management functions are used to interrogate current window attributes.
tim_	Timer-management functions operate on timer objects.
tsk_	Task-management functions allow use of multiple tasks.
win_	Window-management functions operate on window objects.

The assembly language interface actually consists of two interfaces: an object-oriented SEND interface and a non-object-oriented direct CALL interface. SEND is stack based, invoking INT 15 and having both arguments and results passed on stack. CALL is used for "inherently" non-object-oriented tasks and loads its arguments into registers; it uses primarily INT 15H but also INT 21H and INT 10H.

Although in some ways the assembly language interface has more power than the C interface, for sanity and portability reasons this review will concentrate on the C interface. The API's C library implementation is a message-based, object-oriented interface that completely encapsulates system data structures used to implement DESQview objects. Operations are performed on objects almost entirely by sending one of 25 messages to one of 7 different object types. We will define these messages and objects while covering some of the API's most important concepts.

Essential API Concepts

DESQview is a multitasking system that allows many programs to run on their own virtual machines, with each program thinking they have exclusive access to DOS and the hardware—although machine multitasking on a single processor is just an illusion created by giving each program a slice of CPU time.

Each program running on its own virtual machine is called a *process*. A *task* is the part of a process that keeps track of which instruction to execute next and the register contents. Every program initially consists of a single process running a single task, but each process can create more tasks. Tasks are considered lightweight processes: They are less expensive to create and end than processes and have the advantage and responsibility of sharing process address space.

In addition to processes and tasks, DESQview has two other interesting process abstractions not found in other operating systems: the *application* and the *package*. An application is a group of windows consisting of a parent task and all its children. Each parent task is assigned an application number so that the whole group can be referenced at once. A package is a group of applications that cooperate together in some way. Packages allow the user to see one unified program while the developer can elegantly decompose the program among whatever tasks are necessary. This allows true top-down design, thus avoiding kludgy, monolithic programs that must use overlays or manage extended memory.

Window

A *window* is a task's virtual display area. All program output goes to a task's window. Every task is associated with a window and can be considered the same entity.

Intertask Communication

DESQview provides two basic means for tasks to communicate with each other: shared memory and semaphores, and mailboxes. Mailboxes and shared memory are functionally equivalent—nothing can be done with shared memory that can't be done with mailboxes. Even so, each has its proper usage: Shared memory is useful for keeping data structures common to cooperating tasks or for transferring data quickly, while mailboxes are used best in everyday task communication.

Shared memory can be created among several tasks, with each task sharing the same memory address. Each task can independently read and write the same memory area, just like any character array. Obviously, concurrent access by multiple tasks to shared memory can lead to memory corruption. Semaphores (analogous to railroad-crossing semaphores) are used to serialize shared memory access and ensure data integrity.

Mailboxes are implemented with *mailbox objects* and are the primary means of intertask communication used by DESQview. Upon creation, each task has an open mailbox, which is an instance of a mailbox object. Mailbox objects work much like ordinary home mailboxes: Someone who wants to send a message to you must find your address, which is an *object handle* (a unique, 32-bit address assigned by DESQview to each new object), and then mail the message. Mail of arbitrary length and content can be sent by value or reference to a mailbox.

Objects and Messages

As stated earlier, the API uses an object-oriented interface that uses messages to communicate between objects. Each object represents a common programming function, with each function implementation completely encapsulated from the programmer. DESQview itself maintains all data structures related to a function. Traditionally, to use a library function, say, to create a win-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

New
Lower Price
and **Free**
PRO-C Work Bench

"Mary had a 4GL whose performance was very slow and everywhere that Mary went the run-times had to go."

PRO-C
The C Source Code Applications Generator.

Accelerate Your Productivity

Pro-C gives you the greatest gift in the computer world – time. You get high quality, fully commented, error free C source code in a fraction of the time it would take to write it by hand.

Pro-C has always **saved** you time and that **saved** you **money**. Now you **save** even more. Vestronix has dropped the price of Pro-C from $675 US to $399.00! And Pro-C Work Bench, the C source code libraries that let you quickly customize your application, is absolutely FREE! A great program at great savings.

Pro-C will increase your ability to create programs quickly by generating the source code for menus, reports, screens, windows, and multi-file batch updates. Textbook quality C code is written by us, while the elegant system solutions and exciting new algorithms are created by you.

Pro-C looks and feels like a 4GL, but it's not. You can do prototyping, layout, design and generation of applications without learning a proprietary language, needing massive amounts of memory, or ending up with slow running programs. Best of all Pro-C doesn't require any run-time licenses. Finally, a company that treats you like royalty instead of forcing you to pay them.

Pro-C – the programming partner that does the boring, repetitive coding without complaint, pays for itself every time you use it, and doesn't argue with your obviously brilliant program designs.

With PRO-C, everybody's a winner.

Number **287** on the Reader Service Card

PRO-C Order yours to-day. Call 1-800-265-2682 VESTRONIX

COPYRIGHT 1994 BYTE PUBLICATIONS
ELECTRONIC REPRODUCTION PROHIBITED

dow, a programmer declares a window structure (or structures that represent a window), sets various fields within the window structure, and then calls a function passing the window-structure address. In DESQview, a new instance of a window object is created, and an object handle is returned. All further access to that window is made by sending a message to that object telling the object what to do—not by setting fields in a structure and then calling a function. Each object is uniquely identified by its handle.

DESQview knows about seven different objects:

- *Window objects* are the primary data structures manipulated by DESQview. Anything displayed to the screen is displayed in a window. Every task is associated with a window.
- *Keyboard objects* give programs access to keyboard input. They define which window is currently receiving input and how the input is presented to the application.
- *Mailbox objects* are DESQview's means of intertask communication.
- *Panel objects* are a means by which menus, help screens, dialog boxes, and so forth, can be designed, stored on disk, and retrieved by name.
- *Pointer objects* are the program's interface to a pointing device, usually a mouse.
- *Timer objects* allow your program to wait for a specified time interval or for a specified time of day.
- *Objectq objects* are event queues that allow your application to wait for input from several sources.

Messages take the form of function calls that operate on each object type. For a message to use an object, the object must first be created, although some objects are automatically created for each new task. Performing an operation on an object is done by calling a function with the object's handle and necessary data as arguments. Here, for example, is how to create and move a new window:

```
winhan= win_new(title, len_title, rows, cols);
win_move(winhan, row, col);
```

Messages are divided into 12 groups of C library function calls, with each group having a common four-character prefix indicating the type of object the functions operate on or what the functions do. The groups are listed in the box.

Objectq Objects: Event-Driven Programming

A program will have input from many sources, which may include the keyboard, a mouse, a timer, or another task. A program normally becomes aware of input by low-level methods, such as interrupts, by polling different input streams, or by simply paying attention to only one input stream at a time. Through Objectq objects, DESQview collapses all input streams into one event queue.

Event-style programming is common to windowing environments, in which essentially an infinite loop is entered where a task blocks on an event queue. When an input event occurs (for example, a key is pressed, a mouse button is pressed, or a message arrives in your mailbox), an entry is placed on your event queue. Your task unblocks, and you then determine who initiated that event, read the corresponding object queue, and process the event.

Program Examples

Now that we know the *what* of DESQview and its API, let's tie everything together and show the *how*.

DESQview is a rich environment with many features that could be shown. But since many developers are already comfortable with menuing, windowing, and other important yet common programming techniques, we will concentrate on DESQview's methods of process creation, semaphores, and shared memory.

There are three executables, which are provided in the listing accompanying this article: Init (INIT.C) provides a template for process creation; YP (YP.C) is a database process; Shmsem (SHMSEM.C) provides a template for the use of semaphores and shared memory. All three executables use a common library of routines, found in DV_LIB.C. *[A makefile that details how each executable program is made is provided on the Code Disk for this issue. The makefile is not printed in this issue because of lack of space.—Ed.]*

Process Creation

The ability to dynamically create new processes is central to developing a flexible application architecture. DESQview has two methods of creating new processes. A necessary step for both methods is to create a program information file (PIF). The preferred method for creating a PIF is to add a new program through DESQview. Using information supplied by you through two data-entry screens, DESQview creates file *xx*-pif.dvp, where *xx* is a two-letter abbreviation for your program. This file is used to start any program or, in other words, create a new process. A program can be started from DESQview's menu or internally from an application.

Init is a template for how one process would initialize all processes used in an application. When you add Init to DESQview, DESQview prompts for arguments that should be passed to Init. For each program Init should start, enter the full pathname of each program's *xx*-pif.dvp file. Init will automatically append the "-pif.dvp" suffix, and so you need to pass only the two letters for your program.

The heart of Init is function **new_process**, found in DV_LIB.C. All the other code is simply DESQview API initialization and some code to loop through argv, which contains the list of processes to create. To create a new process, an application uses the following call:

```
apphan= app_start(pifbuf, lpifbuf);
ulong apphan;  /* returns application task handle */
char  *pifbuf; /* pointer to PIF string */
int   lpifbuf; /* length of PIF string */
```

Pifbuf is the program's *xx*-pif.dvp file, which for DESQview is 416 bytes long. After reading the 416-byte PIF into pifbuf, **new_process** must modify the PIF default directory and default drive fields to reflect the location of program invocation, thus ensuring that a program's current environment is correct. This is easily

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

done by calling **getcwd** to get the current working directory and then transferring this information into the PIF buffer:

```
pifbuf[100]= default_dir[0];
strcpy(&pifbuf[101], &default_dir[2]);
```

In this example, *default_dir* is the current working directory set by **getcwd**. In the PIF, at byte offset 100, is the default drive, which has a value of A, B, or C (or another appropriate letter), and is the first character in *default_dir*. The default directory name starts at byte offset 101 and is the rest of *default_dir* (excluding the colon). Now pifbuf is set, and a call to **app_start** can be made.

Semaphores

DESQview implements semaphores using mailboxes. To acquire a semaphore, a process locks a mailbox with **mal_lock**. If a semaphore is already locked, then any processes trying to acquire that semaphore will block until the semaphore is released with **mal_unlock**. Notice what must happen for any processes to use a semaphore: Some process must first create the semaphore mailbox with **mal_new** and name the mailbox with a previously agreed-upon name using **mal_name**; then, processes that wish to use the semaphore must get the semaphore mailbox handle using **mal_find**. This is called *attaching* to the semaphore.

Although semaphores work correctly, they are not robust enough to work in a true multiprocess environment. For example, which process first creates the semaphore mailbox? Which process finally removes the semaphore mailbox? DESQview presumes a manager process that first creates the semaphore and removes it as a part of the process exiting. This is not how a multiprocess environment should work—processes should be able to be created and killed and be able to attach to and detach from semaphores at will. Also, a semaphore must be able to be detached from without the mailbox being deleted, as the semaphore may still be used by other processes. This feature requires tracking the number of processes attached to a semaphore.

Process-independent semaphores are implemented through two function calls, **seminit** and **semcmd**, found in DV_LIB.C, and a database process called Yellow Pages (YP). YP is an idea borrowed from Sun, but in a much simplified form. YP is a database-server process that stores records so that processes can share data and dynamically store and look up data. YP also allows users to write server functions that can be called from another task. Only one record type is stored in YP: a YP_MAP, which consists primarily of a key and data field. The YP process is in YP.C; yp_inc.h must be included. YP access routines are in DV_LIB.C.

To create a semaphore, use

```
seminit(key, semflg, semid_p);
```

in which *key* is a predefined string that uniquely identifies a semaphore to all processes that want to use the semaphore. Look in SERVICES.H (Listing 0) for keys used in these examples. Argument *semflg* is a flag that indicates what operation to perform. Choices are found

"To the rescue... the C Toolbox offer a number of solid utilities to reformat, analyze and validate source code...."
---Bill Rubin, The C Gazette, Winter 1988

The C Programmer's Toolbox contains more than 20 tools for the serious C programmer. The tools are available in three different development environments: PC/MS DOS; Macintosh's MPW; and Sun's UNIX. Some of the major tools include:

CDecl	Translates C declaration statements to/from English *(helps you decipher those complicated C variable declarations and construct new ones). {MPW & Sun only}*
CFlow™	Determines how a program is organized and what is in the runtime library *(quickly shows you function hierarchy and interdependancies)*
CHilite™	Highlights and prints your C source code *(great for creating your final documentation). {MPW only}*
CLint™	Syntactically checks one or more of your C source code files *(use this to find those hard to find, syntactically correct programming mistakes)*
Cpp	Preprocesses C source code files that have ANSI macro definitions *(helps you port ANSI C code to/from your target environment). {MPW & Sun only}*
CPrint™	Reformats/beautifies your C source code *(makes code easy to read and ends disputes on source code formats).*
CritPath™	Determine a program's critical path. *(helps you improve your program's performance) {PC DOS only}*
CXref™	Cross references and checks usage of up to 17 different classes of symbols in your C source code *(cross references any size program and number of listings)*
PMon™	Monitors a program's execution *(non-invasively monitors your program's function execution, system service and BIOS usage). {PC DOS only}*

"... The tools I tried operated both quickly and flawlessly..."
---M. Steven Baker, Programmer's Journal, May 1988

C™ Programmer's Toolbox / MPW PC Sun

The tools include support for ANSI C, Microsoft C, Turbo C, Quick C, Sun UNIX C, Macintosh's MPW C, THINK C and more. The Toolbox easily handles any size program through its virtual memory system. Several tools also include a C macro processor that accepts K&R, ANSI and other C dialects.

C Programmer's Toolbox™
MPW ($295), PC Volume I or II ($99.95),
PC Volume I & II ($175), Sun ($495)
C Programmer's McTool Series™ (Macintosh - non-MPW)
McCPrint™ ($59.95), McCLint ($99.95)

"... it's a complete set of utility programs which can make the life of a C programmer easier... I really like this package..."
-- Scott Robert Ladd, Micro Cornucopia, November 1988

Call and Order today...
(CA residents, please add sales tax; Overseas shipping extra)

MasterCard VISA®

MMC AD Systems™
Box 360845
Milpitas, California 95035
(408) 263-0781

Number 94 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

in SHM_INC.H (Listing 0). To create or attach to a semaphore, use *SHM_CRAT*. Argument *semid_p* is a ulong pointer that is set with the semaphore's mailbox handle. When a process calls **seminit**, an entry in YP is created, or the reference count is incremented, indicating that another process is using the semaphore. So, when a semaphore is detached from through function **semcmd**, the reference count in YP is decremented, indicating that a process is no longer using the semaphore. When the reference count reaches 0—indicating no more processes are using the semaphore—the semaphore is removed. Through this method, any process can attach to or detach from a semaphore at any time. All coordination is handled through YP. Acquiring a semaphore is done with **p(semid)**, where *semid* is the value set in **seminit**. To release the semaphore, call **v(semid)**.

Shared Memory

By using **api_getmem**, a process can allocate memory from a shared memory section of system memory. But how do other processes get the address of the shared memory buffer? One way is by mailing the address to all processes that wish to share the memory. The problem with this solution is that it requires knowing the application handle for each task, which in turn requires a manager process to create all the processes and mail the shared memory address to the processes. This again restricts dynamically creating and destroying new processes that need to access shared memory. And when should a shared memory buffer be freed?

As with semaphores, two functions, **shminit** and **shmcmd**, along with YP, are used to create process-independent shared memory. For a process to use shared memory, DESQview requires an asterisk as the first character in the shared program pathname field. This is accomplished in the advanced-options screen presented when adding a new program to DESQview's menu. It is critical that at least 1K of system memory be allocated by each process for shared memory; otherwise, enough memory may not be available.

To create shared memory, call

```
shminit(key, shmflg, shm_pp, shm_len);
```

The arguments are similar to those in **seminit**: *shm_pp* is set with the shared memory address; thus *shm_pp* is a pointer to a character pointer. Argument *shm_len* is the number of bytes of shared memory to allocate. With *shmflg SHM_CRAT*, **shminit** will create the shared memory buffer if it has not already been created or attach to it if it has. When shared memory is created, the address is added to YP with *key* as the key. A process attaching to shared memory actually goes to YP and gets the shared memory address. Detaching is directly analogous to the semaphore case. In both the semaphore and shared-memory library calls, all YP accesses are done in the library calls and are completely transparent.

How It All Fits Together

We now have crafted some powerful tools. We have tools to create new processes easily, tools for these new processes to create cooperatively and independently, and tools to use both semaphores and shared memory. We must now develop a method for using these tools, since their inconsistent use by multiple processes invites some incredibly subtle bugs. Program Shmsem (Listing 0) provides a template of how semaphores and shared memory should be used together. A good test for Shmsem is to make another copy of Shmsem, call it Shmsem1, install it in DESQview, add another argument to Init so that Init creates Shmsem1, and simultaneously run both processes. This will demonstrate that semaphores guarantee exclusive access to shared memory and that shared memory is not corrupted.

Function **test** in Shmsem demonstrates correct usage of Yellow Pages, how to create shared memory and a semaphore, and how to use this semaphore to protect shared memory against corruption. Notice that no error checking is included in the code segments: Every library function always indicates its error status by returning OK (0) for success, or FAIL (-1) for failure.

For display purposes, we get our current task's window handle, winhan, with **win_me**, and to use Yellow Pages, we must first get YP's task handle, yphan, with **yp_open**:

```
winhan= win_me();
yp_open(&yphan);
```

Since we plan to use shared memory, we also must use **seminit** to create a semaphore to protect shared memory from simultaneous access. All processes know this semaphore by the key TEST_SHMSEM. By using *SHM_CRAT*, we chose to create the semaphore if it doesn't already exist or attach to it if it does. Argument *semid* identifies the semaphore for future semaphore operations, much like a file descriptor is used instead of the filename.

```
seminit(TEST_SHMSEM, SHM_CRAT, &semid);
```

Next, we want to create shared memory, but we must first guarantee exclusive access to use shared memory. This is done by acquiring a resource, in this case shared memory, with the **p** operation. Argument *semid* identifies which semaphore we want to use.

```
p(semid);
```

Now we are ready to create shared memory using **shminit**. As in **seminit**, all processes know their common shared memory by key *TEST_SHM*. *SHM_CRAT* is the same as under **seminit**. *Shm_p* is set with the address of the shared memory region. *TEST_SHM_LEN* is the number of bytes of shared memory to allocate.

```
shminit(TEST_SHM, SHM_CRAT, &shm_p, TEST_SHM_LEN);
```

We are, for the moment, finished accessing shared memory. We must release our resource, shared memory, with the **v** operation so that other processes can attach to shared memory. Both **p** and **v** must be atomic operations so that one process is guaranteed to get or release a semaphore at one time. Atomicity is ensured by starting a critical code section with **api_beginc**, locking the

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

mailbox with **mal_lock**, and then ending the critical section with **api_endc**.

```
v(semid);
```

For a sample shared memory operation, we write a string, *shm_string*, *write_cnt* number of times to shared memory. To write to shared memory, we acquire the semaphore with **p**. We then print to our task window, with **win_printf**, shared memory contents; string-copy a different string into shared memory; and, for verification, print our newly copied string. Operation v releases the semaphore, whereupon another process can acquire shared memory.

```
do
{
   p(semid);
   win_printf(winhan, "SHARED MEMORY BEFORE=<%s>\n",
              shm_p);
   strcpy(shm_p, shm_string);
   win_printf(winhan, "SHARED MEMORY AFTER=<%s>\n",
              shm_p);
   v(semid);
} while(--write_cnt);
```

Since we are now done with the semaphore and shared memory, we must detach from them both. First, shared memory is detached, because the semaphore is used to get exclusive access to shared memory. To detach, we call **shmcmd** with the shared memory key, address, and command flag *SHM_DEL*, which informs **shmcmd** to detach from shared memory (**semcmd** is analogous to **shmcmd** in detaching the semaphore):

```
p(semid);
shmcmd(TEST_SHM, shm_p, SHM_DEL);
v(semid);
semcmd(TEST_SHMSEM, semid, SHM_DEL);
```

When no more processes are using the semaphore or shared memory, both will be removed.

Final Thoughts

DESQview, its API, and its constellation of tools form a powerful development environment. Although DESQview may someday be OS/2 compatible, it is not meant to replace OS/2—DOS will be with us for some time. DESQview provides an excellent platform for advanced DOS development while staying compatible with the millions and millions of non-OS/2 systems. Providers of vertical market software and turn-key systems should be especially interested in DESQview. I hope this article has been a useful introduction to DESQview's API.

Todd Hoff is an independent consultant, writer, and instructor. Although Todd is interested in everything, he specializes in UNIX and DOS programming and instruction. Todd hasn't published any books and doesn't belong to any impressive groups, but he is always available for comments. Please feel free to contact him at 1075 W. Remington St., Sunnyvale, CA 94087, 408-739-5707.

Code follows.

C Programmers

Explore a world of DOS memory

Beyond 640K

Use the PowerSTOR C library to exploit:
- LIM 4.0 or LIM 3.2 Expanded Memory
- AT Style Extended Memory
- Hard Disk Space
- Any Combination

- PowerSTOR is the easy way to use system memory resources. It self-configures at run time, under control of your software.
- PowerSTOR makes system memory resources available to your software as STORheaps. STORheaps can be allocated and de-allocated dynamically. Up to a gigabyte of PowerSTOR memory can be used as one huge STORheap or many smaller STORheaps.
- PowerSTOR uses none of the normal heap and it minimizes its usage of stack space. These valuable resources are preserved for your use.

Library with K & R standard source code — only $92.50
(includes shipping)

Order today and explore beyond the limits.

Acme Software

P. O. Box 40734, Portland, Oregon 97240 — Phone (503) 235-2816

Number **305** on the Reader Service Card

Parallel Programming for "C"

INTERWORK™
A Concurrent Programming Toolkit

Interwork is a "C" program library which allows you to write your programs as a set of cooperating concurrent tasks. Very useful for simulation, real-time applications, and experimentation with parallel programming.

FEATURES

- Supports a very large number of tasks (typically more than 100) limited only by available memory. Low overhead per task results in very fast context switching.
- Provides a full set of inter-task communication (ITC) facilities, including shared memory, locks, semaphores, blocking queues, and UNIX™-style signals. Also has building blocks for constructing your own ITC facilities.
- Lets you trace task switches and inter-task communication.

Interwork is available for the following systems:

Hardware	*Operating System*	*Price*
IBM PC, XT, AT	PC-DOS 2.0 or later	$129
IBM PC AT	XENIX™	$159
DEC VAX™, SUN	UNIX 4.2BSD	$249

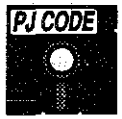
Please specify hardware and operating system when ordering. Shipping and handling included; COD orders add $5.00. Send check or money order to:

Block Island Technologies
Innovative Computer Software
15455 NW Greenbrier Parkway, Suite 210
Beaverton, Oregon 97006
(503) 690-7181 FAX (503) 645-7732

*Trademarks: Interwork, Block Island Technologies; UNIX, AT&T Bell Laboratories, Inc.; XENIX, Microsoft, Inc.; VAX, Digital Equipment Corporation

Number **244** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 88.

Code from "Using the DESQview API," by Todd Hoff

1

```
/*                              COMMON_INC.H
 * Commonly use macros combined in one file. */
#define OK                      0
#define FAIL                    -1
#define FOREVER                 while(1)
#define IS_FLAG_SET(var, flag)  (((var) & (flag)) == (flag))
#define ON_FLAG(var, flag)      ((var) |= (flag))
#define OFF_FLAG(var, flag)     ((var) &= ~(flag))
#define INIT_FLAG(var)          ((var)= 0)
#define MASK(var, mask)         ((var) & (mask))
typedef unsigned char           uchar;


/*                              SERVICES.H
 * General service entry points available to anyone who wants to use them. */
#define YELLOW_PAGES      "YP"              /* Yellow Page Database Service */
#define TEST_SHMSEM       "TEST SHM SEM"
#define TEST_SHM          "TEST SHARED MEM"
#define TEST_SHM_LEN      100


/*                              SHM_INC.H
 * Shared memory include file that contains macros for operations on shared
 *  memory and semaphores. */
#define SHM_CREATE      0x01      /* create operation */
#define SHM_ATTACH      0x02      /* attach operation */
#define SHM_CRAT        0x04      /* create or attach operation */
#define SHM_NOBLOCK     0x08      /* no blocking operation */
#define SHM_DEL         0x10      /* delete operation */


/*                              DV_LIB.C
 *  DESQview library of useful routines:
 *   semaphore, shared memory, process creation, process naming,
 *   Yellow Page, sleep. */
#include <stdio.h>
#include "dvapi.h"
#include "services.h"
#include "common_inc.h"
#include "shm_inc.h"

static ulong   _yphandle= 0;   /* Yellow Page handle- for library use only */

/* Seminit- Create a new semaphore or attach to an already existing one. */
int    seminit(key, semflg, semid_p)
char   *key;           /* predefine unique key naming semaphore for all procs */
int    semflg;         /* flag set with desired options */
ulong  *semid_p;       /* semaphore identifier use for semaphore operations */
{
   YP_MAP   ypmap;      /* Yellow Page map entry */

   /* set YP_MAP with key to semaphore */
   SET_YPMAP(&ypmap, key, strlen(key)+1, NULL, 0);
   if (yp_execute(_yphandle, &ypmap, YP_MAKE_SEM) )
      return FAIL;
   /* set semid_p with new semaphore mailbox handle, from YP */
   memcpy(semid_p, ypmap.data, sizeof(ulong) );

   return OK;

} /* seminit() */

/* Semcmd- general functions to manipulate semaphores. Currently a semaphore
 *  can only be removed. */
int    semcmd(key, semid, cmd)
char   *key;       /* name semaphore to be operated on */
ulong  semid;      /* id of semaphore to be operated on */
int    cmd;        /* cmd to execute on semaphore */
```

2

```
{
   YP_MAP   ypmap;   /* Yellow Page entry buffer */

   switch (cmd)
   {
      case SHM_DEL:
         /* Remove semaphore. We don't actually want to remove a semaphore
          * if other processes are still attached. */
         SET_YPMAP(&ypmap, key, strlen(key)+1, NULL, 0);
         if (yp_execute(_yphandle, &ypmap, YP_DETACH_SEM) )
            return FAIL;
         break;

      default:
         return FAIL;

   } /* switch */

   return OK;

} /* semcmd() */

/* P- the p operation acquires a resource by trying to set the semaphore. The
 * process will block waiting for the resource to free. Only blocking binary
 * semaphores are supported. It would be easy to implement nonblocking counting
 * semaphores. DESQview's mailbox locking mechanism is used. */
int     p(semid)
ulong   semid;   /* semaphore to acquire */
{
   api_beginc();       /* begin critical code section */
   mal_lock(semid);    /* lock mailbox which acts as a semaphore */
   api_endc();         /* end critical code section */

   return OK;

} /* p() */

/* V-  the v operation releases a resource.
 *  DESQview's mailbox semaphore mechanism is used. */
int     v(semid)
ulong   semid;   /* semaphore to release */
{
   api_beginc();         /* begin critical code section */
   mal_unlock(semid);    /* unlock mailbox, release semaphore */
   api_endc();           /* end critical code section */

   return OK;

} /* v() */

/* Shminit- creates or attaches to shared memory. */
int    shminit(key, shmflg, shm_pp, shm_len)
char   *key;           /* Unique name to identify shared memory */
int    shmflg;         /* Flag for shared memory options */
char   **shm_pp;       /* pointer to shared memory address */
int    shm_len;        /* length of shared memory to create */
{
   YP_MAP   ypmap;      /* Yellow Page entry buffer */

   /* set YP_MAP with key to shared memory */
   SET_YPMAP(&ypmap, key, strlen(key)+1, (char *) &shm_len, sizeof(int) );
   if (yp_execute(_yphandle, &ypmap, YP_MAKE_SHM) )
      return FAIL;
   /* set shm_pp with shared memory address, from YP */
   memcpy(shm_pp, ypmap.data, sizeof(char *) );

   return OK;

} /* shminit() */

/* Shmcmd- general functions to manipulate shared memory. Currently a shared
 *  memory pointer can only be removed. */
int    shmcmd(key, shm_p, cmd)
char   *key;        /* name identifying shared memory region */
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
char   *shm_p;       /* pointer to shared memory */
int    cmd;          /* operation on shared memory */
{
   YP_MAP   ypmap;   /* Yellow Page entry buffer */

   switch (cmd)
   {
      case SHM_DEL:
         /* send detach command to YP. set YP_MAP with key */
         SET_YPMAP(&ypmap, key, strlen(key)+1, NULL, 0);
         if (yp_execute(_yphandle, &ypmap, YP_DETACH_SHM) )
            return FAIL;
         break;

      default:
         return FAIL;

   } /* switch */

   return OK;

} /* shmcmd() */

/* Yp_open- gets the Yellow Page process handle. */
int     yp_open(yphan_p)
ulong   *yphan_p;   /* Yellow Page process handle */
{
   if ((*yphan_p= mal_find(YELLOW_PAGES, strlen(YELLOW_PAGES))) == 0)
      return FAIL;
   _yphandle= *yphan_p;       /* set internal YP handle */

   return OK;

} /* yp_open() */

/* Yp_execute- sends a server command to YP and receives the response. */
int         yp_execute(yphan, ypmap_p, cmd)
ulong       yphan;       /* YP process address */
YP_MAP_P    ypmap_p;     /* YP MAP entry to detach */
int         cmd;         /* server command to in YP */
{
   int
      ypmap_len,        /* length of return YP_MP entry */
      status;           /* status of detach command: OK or FAIL */
   YP_MAP_P
      rcv_ypmap_p;   /* pointer to entry sent back by YP */

   if (ypmap_p == NULL)    /* no data to send */
      mal_addto(yphan, ypmap_p, 0, cmd);
   else
      mal_addto(yphan, ypmap_p, sizeof(YP_MAP), cmd); /* data to send */
   /* if command succeeded and data sent back, copy this user's buffer */
   if ((status= mal_read(mal_me(), &rcv_ypmap_p, &ypmap_len)) == OK
      && ypmap_len == sizeof(YP_MAP) )
   {
      memcpy(ypmap_p, rcv_ypmap_p, sizeof(YP_MAP) );
   }

   return status;

} /* yp_execute() */

/* Sleep- put process to sleep for number of seconds.
 * Implemented using DESQview's timer object. */
int     sleep(seconds)
ulong   seconds;       /* number of seconds to sleep */
{
   ulong   timhan;   /* handle for timer */

   timhan= tim_new();   /* create timer instance */
   tim_addto(timhan, seconds*100);   /* DV uses 1/100 of second */
   tim_read(timhan);       /* block until timer "buzzes" */
   tim_free(timhan);       /* deallocate timer */
```



```
   return OK;

} /* sleep() */


#define PIF_LEN                416
#define PIF_DEFAULT_DRIVE_LOC   100
#define PIF_DEFAULT_DIR_LOC     101
#define PIF_MAX_DIR_NAME_LEN     64
#define PIF_COMMON_PATH_NAME    "-pif.dvp"

/* New_process- creates a new process to run under DESQview. Every program under
 *  DV has a PIF (program information file) that describes the program. These
 *  files end in "-pif.dvp" and are created by DV. "-pif.dvp" should
 *  not be appended to the process name. It is the PIF file not the executable
 *  file name that is needed to create a process. */
int     new_process(proc, newhan_p)
char    *proc;       /* name of process to create: -pif.dvp file */
ulong   *newhan_p;   /* handle of created process */
{
   FILE
      *fp,          /* -pif.dvp file pointer */
      *fpopen();
   char
      proc_dvp[PIF_LEN],    /* buffer for process -pif.dvp file */
      defdir_name[PIF_MAX_DIR_NAME_LEN];   /* buffer for full pif pathname */

   /* form full -pif.dvp filename */
   sprintf(defdir_name, "%s%s", proc, PIF_COMMON_PATH_NAME);
   if ((fp= fopen(defdir_name, "rb")) == NULL) /* open pif file */
      return FAIL;

   if (fread(proc_dvp, PIF_LEN, 1, fp) < 1)   /* read in whole pif file */
      return FAIL;

   /* get current directory to set processes current working directory */
   if (getcwd(defdir_name, PIF_MAX_DIR_NAME_LEN) == NULL)
      return FAIL;

   /* assign default drive location into pif file buffer */
   proc_dvp[PIF_DEFAULT_DRIVE_LOC]= defdir_name[0];
   /* then copy the rest of the default drive into pif file buffer */
   strcpy(&proc_dvp[PIF_DEFAULT_DIR_LOC], &defdir_name[2]);

   /* start new process and assign process handle */
   if ((*newhan_p= app_start(proc_dvp, PIF_LEN)) == 0)
      return FAIL;

   return OK;

} /* new_process() */

/* Name_process- uniquely names a process by searching for a mailbox with
 *  processes name. An already existing mailbox by that name is an error. If
 *  no error then process is named. */
int     name_process(process_name)
char    *process_name;
{
   int   rc= FAIL;   /* function return code */

   api_beginc();
   /* find if process already exists */
   if (mal_find(process_name, strlen(process_name)) == 0)
   {
      /* if it doesn't then name it */
      mal_name(mal_me(), process_name, strlen(process_name) );
      rc= OK;
   }
   api_endc();
   /* if process does exist then its a duplicate and function fails */
   return rc;

} /* name_process() */
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Hoff code, continued

5

```
/*                              INIT.C
 * Create all processes given on command line, run from DESQview menu.
 * init: process_to_create_names
 *   process_to_create_names: are full path names to filename-pif.dvp files.
 *   These files are created by ADDING a program to DESQview. DO NOT give
 *   executable filenames. Filename is the two-character abbreviation given for
 *   programs added to DESQview. For DESQview argument space reasons do not use
 *   -pif.dvp suffix; this is assumed and automatically appended. */

#include <stdio.h>

#include "common_inc.h"
#define API_REQ_VER   0x201      /* required API version for this program */

main(argc, argv)
int   argc;
char  **argv;
{
   if (api_init() < API_REQ_VER)       /* DESQview API initialization */
   {
      fprintf(stderr, "Program requires DESQView version %d.02%d or later.\n",
         API_REQ_VER/256, API_REQ_VER%256);
   }
   else
   {
      api_level(API_REQ_VER);    /* set API version number */
      init_processes(--argc, &argv[1]);    /* create all processes */
      api_exit();           /* restore normal control code processing */
   }

} /* main() */

/* Init_processes
 * Takes a list of processes and creates them using DESQview routines. */
int   init_processes(no_proc, proc_lst)
int   no_proc;          /* number of processes to create */
char  **proc_lst;       /* list of processes to create */
{
   char   **end_proc_lst;    /* last process to create */
   ulong  newhan;            /* handle of new process */

   end_proc_lst= &proc_lst[no_proc];   /* set end of list */
   /* for each process in the list */
   for ( ; proc_lst != end_proc_lst; proc_lst++)
   {
      if (new_process(*proc_lst, &newhan) ) /* create the new process */
         return FAIL;
   } /* for */

   return OK;

} /* init_processes() */


/*                              SHMSEM.C
 *
 * shmsem: test_shm_string write_count
 *   test_shm_string: is string that process writes into shared memory
 *   write_count: number of times to test shared memory
 *
 * Template for shared memory and semaphore use. */

#include <stdio.h>

#include "services.h"      /* system level public services */
#include "shm_inc.h"       /* shared memory/semaphore macros */

#include "common_inc.h"
#define API_REQ_VER   0x201      /* required API version for this program */

main(argc, argv)
int    argc;
char   **argv;
```

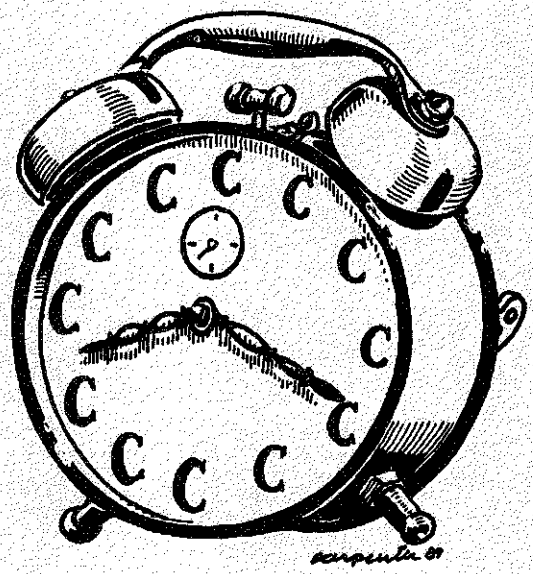
6

```
{
   if (argc < 3)   /* validate number of passed arguments */
   {
      fprintf(stderr, "usage: test \"shared mem string\" write_count");
      exit(1);
   }

   if (api_init() < API_REQ_VER)   /* do DESQview API initialization */
   {
      fprintf(stderr, "Program requires DESQView version %d.02%d or later.\n",
         API_REQ_VER/256, API_REQ_VER%256);
   }
   else
   {
      api_level(API_REQ_VER);      /* set API version number */
      test(argv[1], atoi(argv[2]) ); /* start test */
      api_exit();          /* restore normal control code processing */
   }

} /* main() */

/* Test- runs test of shared memory and semaphores */
int   test(shm_string, write_cnt)
char  *shm_string;   /* string to write into shared memory */
int   write_cnt;     /* number of times to write string into shm */
{
   ulong
      winhan,      /* handle of my task's window */
      yphan;       /* handle of Yellow Page process */
   char
      *shm_p,      /* pointer to created or attached shared memory */
      *semid;      /* semaphore identifier */

   winhan= win_me();       /* get my task's window handle */
   if (yp_open(&yphan) )   /* get handle to Yellow Page server */
      return FAIL;

   /* initialize semaphore- may create or attach to semaphore */
   if (seminit(TEST_SHMSEM, SHM_CRAT, &semid) )
      return FAIL;

   p(semid);   /* acquire semaphore for shared memory processing */
   /* attach to or create shared memory */
   if (shminit(TEST_SHM, SHM_CRAT, &shm_p, TEST_SHM_LEN) )
      return FAIL;
   v(semid);   /* release semaphore when done */

   do   /* write_cnt number of times */
   {
      p(semid);   /* acquire semaphore for writing into shared memory */
      win_printf(winhan, "SHARED MEMORY BEFORE=<%s>\n", shm_p);
      strcpy(shm_p, shm_string);
      win_printf(winhan, "SHARED MEMORY AFTER=<%s>\n", shm_p);
      v(semid);   /* release semaphore when done */

   } while(--write_cnt);

   p(semid);   /* acquire semaphore to detach from shared memory */
   if (shmcmd(TEST_SHM, shm_p, SHM_DEL) )
      return FAIL;
   v(semid);   /* release semaphore when done */
   if (semcmd(TEST_SHMSEM, semid, SHM_DEL) )   /* detach from semaphore */
      return FAIL;

   return OK;

} /* test() */
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Computing Time in C

by
Murray L. Lesser



No rational programmer, who knows anything about how the real world measures time, would use the time and date functions declared in the ANSI C standard library time.h file, except under duress. The "standard" function definitions are based on the erroneous notion that "time interval" can be calculated from "calendar time." To add insult to the error, when running under DOS, each use of a time.h function includes the redundant recalculation of a simulated UNIX system clock from the existing DOS time and date functions.

Fortunately, most DOS C compilers furnish the more-rational DOS-based time and date functions in addition to the clumsy ANSI C and UNIX V time-related functions. Unfortunately, the function names and data structures used for these DOS-based functions vary between implementations. In what follows, the examples are for Borland's Turbo C, version 2.0, but can be transliterated into most other DOS C library dialects.

In this article, I describe how the real world measures time, and why the DOS system clock is a good approximation to reality. There is a discussion of the mistake made by the implementers of DOS 3.3 with the "enhancement" that writes to the real time clock every time a program writes to the DOS clock, along with a procedure for patching your copy of DOS 3.3 (or later) to cut this ill-conceived cord. There are two assembled modules for Turbo C to allow direct reading from, and writing to, the real time clock. This first of two articles on keeping time by computer ends with a demonstration C program to reset the real time clock to agree with changes in time zone—such as at the start and end of Daylight Savings Time.

Plauger and Brodie's statement . . . is probably a fair indication of how little the ANSI C Language Standards Committee understood the art of keeping time.

Defining Time

Before discussing the subject of how time is measured in computer systems, it might be well to spend a few words discussing how the real world measures time, and what kind of time is being measured. There is a semantic problem because the word "second" is used in both "calendar time" and "time interval," although the relationship between these two kinds of time is tenuous, at best. There is a further source of confusion because the basis of civil (calendar) time was changed from Greenwich Mean Time (GMT) to a redefined Coordinated Universal Time (UTC) on January 1, 1972.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

There are always 86,400 "mean solar seconds" in a "mean solar day" in GMT, but the length of the day, and hence of the second, depends on the average rotational rate of the Earth on its axis. This rate varies slightly, in a manner that is not predictable. Thus, the only way to measure GMT accurately is by astronomical observation—usually of the meridian transit of a fixed star. Prior to the invention of the atomic clock, such observation was the most accurate time-keeping technology available.

GMT, as used by astronomers, is known as UT1. (Astronomers also use two other Universal Times, UT0 and UT2, which are not of great interest here.) Coordinated Universal Time (UTC) was first devised by the U.S. Naval Observatory and the Royal Greenwich Observatory for standard time broadcasts based on atomic-clock time. This first UTC included a frequency offset from the atomic second, predicted annually, intended to keep the time broadcasts as close as possible to the "observed" UT2. Since the best predictions of the Earth's actual rotational rate are not very good, it was necessary to adjust the origin of the UTC time base periodically, in 100 millisecond steps, to keep broadcast time in reasonable agreement with time as measured by the rotation of the Earth. This version of UTC was in use from 1961 through 1971.

It was eventually realized that the "sliding time scale" of the old UTC was not really practical, so Coordinated Universal Time was redefined. Since January 1, 1972, Coordinated Universal Time differs from GMT (UT1) in that each second in UTC has the same duration—the International Standard (SI) second—but not all UTC days have 86,400 seconds. (The SI second is the unit of measurement used for time intervals. It is related to calendar time only by the 1972 definition of UTC.)

Plauger and Brodie's statement in *Standard C* (see references) describing the standard library function gmtime as " . . . expressed as Universal Time Coordinated (UTC, formerly GMT) . . . " is probably a fair indication of how little the ANSI C Language Standards Committee understood the art of keeping time. If you've read this far, you know that the initials UTC do not stand for "Universal Time Coordinated" and that UTC is not a new name for GMT.

Although the redefined Coordinated Universal Time has replaced GMT as the basis for civil time, GMT is still in use for other purposes. The international standard for UTC requires that, at any instant, the time of day measured on a UTC clock must be in agreement with the time of day measured on a GMT clock within 0.9 second (usually, UTC is held within 0.7 second of GMT). The actual difference between UT1 and UTC, in tenths of a second, is coded into all national standard time broadcasts every minute. As I write this, GMT is 0.4 second slow, relative to UTC.

So far, GMT seconds have been longer than UTC seconds, so the requirement for matching clocks has been met by inserting extra seconds into the UTC year. The Bureau Internationale de l'Heure is responsible for deciding when a "leap second" is needed. The Bureau is authorized to insert a leap second (either positive or negative) as the last second of any UTC month, but gives preference to December and June. There were two leap seconds inserted into 1972—when getting the new UTC under way. The years since have had either only one, or none. There is no way to predict when a leap second will occur; national time keeping organizations are given only a few weeks notice.

Now, as is known to every navigator who has studied the "Time" chapter in *Bowditch*, it is usually impossible to calculate a time interval by the difference between two calendar times. The time interval corresponding to each calendar second prior to January 1, 1972 is not a constant; nor is the number of seconds in each calendar day after that date. The ANSI C standard library function difftime is not computable unless both calendar times are in the same UTC month, later than January 1, 1972. I suppose one could determine the value of difftime by table look-up, if neither date involved were more recent than when the compiler was updated, but I don't know of any C compiler that goes to that trouble.

The DOS System Clock

The DOS system clock is based on two counters in memory. Time of day is computed from the PC timer counter, a long integer at address 40:6C. The date counter is maintained by DOS, in an undocumented version-specific memory location (in PC-DOS version 3.3, the DOS date counter is a word at address 70:05DA).

The PC timer runs at a nominal input frequency of 1.19318 MHz with a divisor of 65536. If you do the arithmetic, you will find that the output (interrupt 0x08) occurs 1,573,040 times per 24 hours, or slightly more often than 18.2 "ticks" per second. At each interrupt, the low-order word of the timer counter is incremented and tested for overflow. If overflow has occurred, the high-order word is also incremented. Then, both words are tested for the maximum allowable value, 0x1800b0. If this has been reached, both words are reset to zero and the adjacent flag byte is incremented—to inform DOS that the timer counter has crossed its midnight.

BIOS support for the DOS (or any other) system clock is built into interrupt 0x1a. Interrupt 0x1a, AH=0, returns the value of the timer count in registers CX:DX, and the value of the overflow flag in register AL. It also resets the overflow flag to zero. Using the interrupt with AH=1 loads the values in registers CX:DX into the timer counter, and also resets the flag byte to zero.

When DOS reads the timer counter with interrupt 0x1a, it uses the value returned in AL to increment its date counter, which contains the number of days elapsed since January 1, 1980. It then converts the timer count value to a four byte internal structure giving the components of a 24-hour clock to a precision of .01 second. This value is returned to the program in the processor registers if the timer count was read as a result of a DOS "get time" request (DOS function 0x2c). It is thrown away if the timer count was read as a result of "get date" (function 0x2a) or of the periodic "read timer value" requests COM-

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

MAND.COM (DOS 3.3 or later) issues while it is waiting for a keystroke input to its command line.

Even though DOS calculates time-of-day to a precision of .01 second, the value returned cannot be more accurate than the tick rate, approximately every 55 milliseconds.

If the timer counter was initialized correctly on system boot, if it has not been reset by any program since, and if the timer clock is actually running at its nominal frequency, timer counter midnight and civil time midnight will occur at the same instant—unless there happened to be a leap second that day. The DOS "get time" function is the best short-term approximation to local civil time that a PC can provide.

If the DOS date counter was set correctly on system boot, and if the timer counter midnight corresponds to civil midnight, the DOS "get date" function will always return the current local date—unless a program has used BIOS interrupt 0x1a to read the timer counter (thereby clearing the overflow flag) between midnight and the next time DOS used that same function. In that case, the DOS calendar will lose a day.

So, it is important that a PC program not use BIOS interrupt 0x1a with AH=0 if there is any chance the program will be run near midnight. (The fastest safe way to read the timer counter is with an assembled function, inhibiting external interrupts while both words are moved into the processor registers.) Unfortunately, the standard C time.h library includes a built-in booby trap tempting implementers of C compilers to violate this precept, and the people who built Turbo C version 2.0 fell into it.

According to Plauger and Brodie, the clock function is defined to return " . . . the number of clock ticks of elapsed processor time, counting from a time related to program startup . . . " If taken literally, this means the timer counter should always be read during program startup and the value stored as an origin in the event the program calls clock. Turbo C, version 2.0, implements clock in the obvious way with interrupt 0x1a, making it dangerous to start any Turbo C program at the witching hour. Microsoft C, version 5.1, avoids the trap but cheats. It evaluates clock using its version of the UNIX V ftime function, which wastes the processing time to approximate the noncomputable time interval since 00:00:00 GMT January 1, 1970, to the millisecond—simulating a wholly fictitious tick rate of once per millisecond.

DOS and the Real Time Clock

Any installed real time clock is hardware independent of the DOS system clock. Factory-installed real time clocks are supported by BIOS functions. You *should* be able to write your own system utilities that reset the system clock (say, to change the creation date of a volume label) and then put your system back on the correct time by resetting it from the real time clock before exit, using the BIOS function to read from the real time clock (there is no equivalent DOS function available to the programmer). You **could** do this under DOS versions prior to PC-DOS 3.3.

The first notice DOS paid to the possible existence of a real time clock was with PC-DOS version 3.0, introduced with the PC/AT. This version of IBMBIO.COM had throwaway code (written over after execution) using values from the real time clock to initialize the BIOS timer counter and the DOS date counter at boot time. PC-DOS 3.2 added code to retain the "throwaway" code if the running system was a PC Convertible, thus allowing the timer and day counters to be reinitialized when the Convertible's power was turned on in "resume" (warm boot) mode.

Both of these enhancements were useful, and did no damage to the intrinsic separation of the two clock systems.

All was well until someone at Microsoft or IBM noticed that DOS did not provide the programmer with an easy way to reset the real time clock. It was decided to remedy the situation by "enhancing" PC-DOS version 3.3. The poorly thought out solution to a nonexistent problem was to reset the real time clock (both time and date) to match the system

Confusing Code?

```
#include <stdio.h>
text_count(){int c,nlines,nwords,nchars,inword;
inword=NO;nlines=nwords=nchars=0;while((c
=getchar())!=EOF){++nchars;if (c=='\n')
++nlines;if ((c==' ')||(c=='\n'))inword=NO;else
if(inword==NO){inword=YES;++nwords;}}printf(
"%d %d %d\n",nlines,nwords,nchars);}
```

C It Your Way!

```
#include <stdio.h>

text_count ()
{
    int c, nlines, nwords, nchars, inword;

    inword = NO;
    nlines = nwords = nchars = 0;
    while ((c = getchar()) != EOF) {
        ++nchars;
        if (c == '\n')
            ++nlines;
        if ((c == ' ') || (c == '\n'))
            inword = NO;
        else if (inword == NO) {
            inword = YES;
            ++nwords;
        }
    }
    printf ("%d %d %d\n", nlines, nwords, nchars);
}
```

NOW with **C-Clearly™**, format C source code exactly the way you want it. C-Clearly's context sensitive analysis will format any C program in your own personal or corporate style.

EASY to use, C-Clearly's style templates are a snap to modify, since they resemble C source code you edit into your preferred format. Templates are included for several common styles as well as standard K & R.

LISTINGS can also be created with function names and comments highlighted for improved readability. Listing options include line numbers, headers and/or footers and flow lines.

IDEAL for making obtuse code clear. Allows all of your source code to be presented in a consistent format of your choosing. Also great for code walkthroughs and final documentation listings.

WORKS with all IBM PC, XT, AT, PS/2 and compatibles, with 512K RAM. Automatically processes all include files and pre-processor statements. ANSI-C compatible. Not copy protected.

C-Clearly **$129.95**

Shipping & Handling USA $3; Canada/Mexico $10; Other Countries $15; CA Residents add sales tax. Visa/MasterCard/COD accepted.

For orders or information call:

1-800-662-8266

V COMMUNICATIONS, INC.
3031 Tisch Way, Suite 802, Dept. PJ5
San Jose, CA 95128 (408) 296-4224

Number 341 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

clock, whenever either the system time or date is reset.

This ill-considered move invalidates many home-grown system utilities and introduces clock errors when running many proprietary programs. The proprietary programs that depend on the independence of the two clock systems reset the system clock to new values (for operational reasons) during execution, and then try to put it back to the calculated "correct" time at the finish. If the system clock is reset many times during the session (Microsoft's Flight Simulator 3 is a particularly egregious example of this), attempting to restore the correct time by calculation doesn't work. The PC-DOS CLOCK$ routine truncates, rather than half-adjusts, when converting time-of-day to ticks. The effect is to lose one tick almost every time DOS is used to reset the system clock. Prior to DOS 3.3, this error was corrected automatically on every system boot. But under DOS 3.3 as distributed, this error became cumulative; carried in the real time clock until it is large enough to be noticed.

If you are running under DOS 3.3 (or later), the best solution to the problem created by the DOS implementers is to patch your own DOS CLOCK$ driver, removing the hazard at its source. Start the patching process with a DISKCOPY of your distribution DOS diskette and apply all the vendor-furnished bug patches to it. Then remove the "read only/hidden/system" attribute bits from the directory entry of the first file. The easy way to do this is with the Norton "File Attribute" utility, but you can write a small program in Turbo C using the **_chmod** (DOS, not UNIX) function to change the file attribute to zero. If you are using PC-DOS, the name of the file is IBMBIO.COM. If you are using a compatible version of MS-DOS, the name of the file is likely to be IO.SYS.

Once the DIR command shows this file, load the file into a simpleminded debugger. I use SYMDEB, but DEBUG will do. Use the debugger "Search" command to find the two-instruction sequence in the CLOCK$ driver that writes the time to the real time clock. The first will be either **MOV AX,0300** (used by PC-DOS 3.3) or **MOV AH,03**, and the second will be **INT 1A**. The corresponding "search" byte argument is either "B8 00 03 CD 1A" or "B4 03 CD 1A." Whichever you find, check by disassembling four instructions from the address one byte before the sequence was found. The first of the four should be **CLI** to inhibit external interrupts, the next sets AX to 0x0300 (or AH to 0x03), the third calls interrupt 0x1a, and the last should be **STI** to re-enable external interrupts. This instruction sequence does half the dirty work by resetting the real-clock time.

The easy way to delete this code is to replace the five (or four) bytes between the **CLI** and the **STI** with **NOP**s. The elegant way is to disassemble prior instructions, starting about 0x30 bytes before the AX (or AH) register was loaded. You are looking for a **CMP Byte Ptr [xxxx],00** followed by a **JZ yyyy**.

The Software Engineering Store

C

Borland Turbo C v2 Prof. $175
Blaise C Tools Plus/6.0 ..... $119
Clear Clear Plus for C $169
Creative Vitamin C $169
Genus PCX Prgmr Toolkit $155
manipulate ".pcx" graphics files
Gimpel PC Lint $99
Greenleaf Comm Library $179
Island Systems C Menu $129
Microsoft C v5.1 ..... $325
Microsoft Quick C $73
Novell Btrieve $199
Sequitor Codebase 4 ..... $249
read, write & create dBase III and Clipper files
Wordcraft C Workshop $69
C training manual & software

Other Tools

Digitalk Smalltalk/286 ..... $164
Burton Tlib $89
source code management system
California Scientific Brainmaker $159
neural network development system
Genus PCX Toolkit $155
Helios Proteus $169
Matrix Layout v2 ..... $175
Microsoft Macro Assembler $111
Microsoft Windows SDK v2.1 ..... $349
Mortice Kern MKS Toolkit v3 ..... $219
KORN shell & 130 UNIX™ -like utilities
Periscope I 512K v4 ..... $499
Qualitas 386-Max $69
80386 memory manager
Quarterdeck API Ref & C Library .. $179
Solution Systems Brief ..... $call
Turbo Power Turbo Professional $99
Whitewater ACTOR $449
High productivity development for MS Windows

C++ Version 2

Zortech C++ is the only true C++ compiler available for MSDOS. Now includes AT&T C++ v2 features including multiple inheritance and type safe linking. C++ debugger lets you view objects instead of translated C code.
C++ v2 ..... $175
C++ Debugger $129
C++ Developer's Edition $399

The Software Engineering Store

1-800-443-7176

In Calif: (415)957-0111

Visa • MasterCard • AMEX
COD • Check • P.O.

- Most orders shipped within 24hrs
- Free shipping on orders over $100

657 Mission Street, #300-PJ
San Francisco, CA 94105

Free Newsletter

Call or write for a catalog and your free subscription to our newsletter.

Prices subject to change without notice. ©1989 TSE. All Rights Reserved

Number 98 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The yyyy should be the offset of the instruction following the **STI** you previously disassembled. If you find it, you can change the **JZ yyyy** to **JMP SHORT yyyy** and save the processing time associated with the data conversion routine.

Now go through the same process, looking for **MOV AH,05** (or **MOV AX,0500**) followed by **INT 1A**. This is the pair of instructions that resets the real time clock date. Eliminate them in a manner equivalent to that used for the time.

Write the whole modified file back to its diskette, change the file attribute back to "read only/hidden/system" (attribute 7), and the deed is done. After testing your handiwork, you can replace the equivalent file on your working DOS disk by using the DOS "SYS" utility.

Controlling the Real Time Clock

The routines shown in the first two listings allow the real time clock (both time and date) to be read from, or written to, with C functions analogous to the DOS-based system clock functions. They are system dependent in two respects. First, the data structures used for time and date values are those declared in the Turbo C, version 2.0, dos.h file. If you are using some other DOS compiler, check its dos.h file for the time and date structures it uses, and modify the two modules accordingly.

Second, and somewhat more difficult to circumvent, is the dependence on a factory-installed real time clock, compatible at the BIOS level with that originally installed in the PC/AT. If you are using an add-in clock, you will have to rewrite these two modules rather completely, based on the input/output register controls required by your clock. The necessary information should be available from the vendor of your clock.

The two functions reading the time and date from the real time clock are in the module CLKREAD.ASM (Listing 1). The **clktime** function loads the time obtained into the memory locations corresponding to the time structure passed as a pointer argument. Since the granularity of the real time clock is one second, **clktime** loops internally until BIOS has returned a different value in the "seconds" register. Thus, **clktime** always returns to the calling program at the same fraction of the real time second. The function returns a zero to indicate a success. If BIOS finds the clock inoperative, or if there is no real time clock installed, the function returns -1 and leaves the argument time structure unchanged.

The **clkdate** function reads the real time clock date. It returns similar error indicators, but it doesn't wait for the start of a new day.

The equivalent functions to set the real time clock, **timeset** and **dateset**, are in the module CLKSET.ASM (Listing 2). The **timeset** function can not set the real time clock to an exact second. Changing the values in the clock registers (setting the clock) does not change the clock update cycle time, so the clock registers continue to be updated at the same actual time each second. Thus, the real time clock usually will be slightly fast (as read by **clktime**) relative to the system clock if both were set to the same "exact" second. It is possible to measure this setting "error" to within about 0.1 second if there is any need to do so.

These modules are written to interface with Turbo C programs compiled for either small or tiny memory models. I use Microsoft's LIB library manager to maintain a PCS.LIB file containing all of my private function modules. My private.h header file contains the corresponding function prototypes.

The demonstration program ZONCH.C (Listing 3) uses the "time" functions in the two modules to reset the real time clock to reflect an actual change in time zone. (In DOS C compilers, the UNIX function **tzset** affects only an apparent change in time zone; it does not reset any clocks.) I wrote the original version of ZONCH on my PC Convertible while waiting between flights at Heathrow Airport, but its major use is to reset the clocks on my systems at the start and end of Daylight Savings Time.

ZONCH reads the real time clock time, adds the time zone increment entered as a command-line argument, and then resets the real time clock.

It will not reset the real time clock across midnight, to avoid calendar troubles.

As shown in Listing 3, ZONCH will run correctly under DOS 3.3 even without the patch described above, since it doesn't reset the system clock (you will have to reboot). If you are using an earlier version of DOS, or have patched your version 3.3 (or later), you can safely add code to reset both the time and date on the system clock from the real time clock. The hazard in resetting the system clock with an unpatched DOS 3.3 occurs around midnight, local time before the change. Given the probable accuracy of the two clock systems, it is unlikely that the system clock and the real time clock will cross midnight at the same actual moment, so the two date counters may not agree when the program is run. PJ

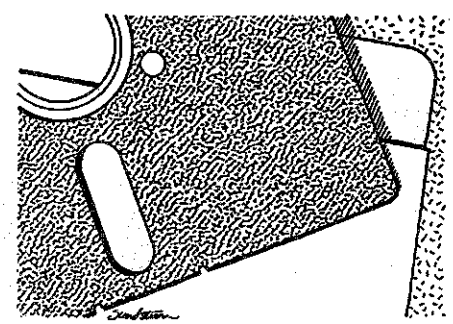
Murray L. Lesser is a retired IBM engineer. In addition to having had several articles published in the small computer press, Murray has written three books dealing with enhancing various Microsoft BASIC compilers with assembled modules. The most recent of these, Advanced QuickBASIC 4.0, *was published by Bantam last year.*

Code follows.

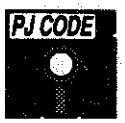
References

Plauger, P. J., and Brodie, Jim. *Standard C*. Redmond, Wash.: Microsoft Press, 1989.

American Practical Navigator (Bowditch), Vol. 1, Defense Mapping Agency Pub. No. 9 (1977 edition).



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 88.

Code from "Computing Time in C," by Murray L. Lesser

1

```
LISTING 1
        page    ,105
        title   CLKREAD
        .model  small
;*************************************************************************
;                               CLKREAD.ASM                               *
;       A pair of assembled functions for linking to Turbo C tiny or      *
;    small memory-model programs to read the installed BIOS real-time     *
;    clock.  The time and date data structures used are declared in the   *
;    Turbo C dos.h header file.                                           *
; int clktime(struct time *t)                                             *
;    reads the real-time clock time until the start of a new second.      *
;    It then loads the time into the time structure, returning a zero     *
;    if the process was successful.  clktime() returns -1 if no BIOS      *
;    real-time clock is installed or an installed clock is inoperative.   *
; int clkdate(struct date *d)                                             *
;    reads the date from the BIOS-defined real-time clock, putting the    *
;    values so obtained into the date structure, returning a zero if      *
;    the read is successful.  clkdate() returns -1 if no BIOS real-time   *
;    clock is installed or an installed clock is inoperative.             *
;                    Written by M. L. Lesser, 7-29-88                     *
;            Assembled with Microsoft MASM v 5.1, switch /MX              *
;*************************************************************************
.code

convert:                ;Local subroutine to convert bcd in AL to hex
        mov     ah,al           ;Separate digits in two registers
        and     al,0fh          ;Keep low-order digit in AL
        shr     ah,1            ;Shift high-order digit to low nibble
        shr     ah,1            ;  of AH
        shr     ah,1
        shr     ah,1
        aad                     ;Convert unpacked bcd to hex
        ret

        public  _clktime
_clktime proc           ;Read clock and fill structure
        push    bp              ;Save significant registers
        mov     bp,sp
        push    di
        push    ds              ;Establish ES addressability to
        pop     es              ;  DGROUP
        mov     di,4[bp]        ;Pointer to time structure (small model)
        mov     ch,0ffh         ;Dummy to check if clock is installed
        mov     ah,02           ;Read real-time clock time
        int     1ah
        mov     ax,0ffffh       ;Dummy output for invalid return
        jc      done            ;Installed clock inoperative
        cmp     ch,23h          ;Maximum valid value for hours (23 bcd)
        ja      done            ;Clock not installed
        mov     bl,dh
        cld
start:  mov     ah,02           ;Compare values of successive seconds
        int     1ah             ;  readings until a change indicates
        cmp     dh,bl           ;  the start of a new second
        jz      start
        mov     al,cl           ;Minutes (bcd)
        call    convert
        stosb                   ;Store in time structure
        mov     al,ch           ;Hours
        call    convert
        stosb
```

2

```
        xor     al,al           ;Dummy hundredths
        stosb
        mov     al,dh           ;Seconds
        call    convert
        stosb
ok:     xor     ax,ax           ;Zero return for valid result
done:   pop     di
        pop     bp
        ret
_clktime endp

        public _clkdate
_clkdate proc
        push    bp              ;Save significant registers used
        mov     bp,sp
        push    di
        mov     ax,ds           ;Set es addressability to dgroup
        mov     es,ax
        mov     di,4[bp]        ;Address of argument structure
        mov     ch,0ffh         ;Dummy output to check return
        mov     ah,04h
        int     1ah             ;Read clock date
        mov     ax,-1           ;Invalid "read" error symbol for output
        jc      done            ;Clock inoperative symbol from ROM BIOS
        cmp     ch,20h          ;Check for "no clock installed" case
        ja      done
        cld
        mov     al,ch           ;Convert bcd centuries to hex
        call    convert
        mov     bh,100
        mul     bh              ;Centuries in AX
        mov     bx,ax           ;Save for later
        mov     al,cl           ;Convert year in century
        call    convert
        add     ax,bx           ;Year
        stosw                   ;Store it in date structure
        mov     al,dl           ;Day
        call    convert
        stosb                   ;Store byte in structure
        mov     al,dh           ;Month
        call    convert
        stosb
        jmp     ok
_clkdate endp
        end


LISTING 2
        page    ,105
        title   CLKSET
        .model  small
;*************************************************************************
;                               CLKSET.ASM                                *
;       A pair of assembled functions for linking to Turbo C tiny or      *
;    small memory-model programs to set the installed BIOS real-time      *
;    clock.  The time and date data structures used are declared in the   *
;    Turbo C dos.h header file.                                           *
; void timeset(struct time *t)                                            *
;    sets the BIOS defined real-time clock time.                          *
; void dateset(struct date *d)                                            *
;    sets the BIOS defined real-time clock date.                          *
;       No checks are made for the existence of a real-time clock, nor    *
;    for valid input times or dates.                                      *
;                    Written by M. L. Lesser, July 23, 1988               *
;            Assembled with Microsoft MASM v. 5.1 , switch /MX            *
;*************************************************************************
.code
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
convert:                ; Local subroutine converts hex in AL to bcd
        aam                     ;Divide AL by 10
        shl     ah,1            ;Move high digit in AH to left
        shl     ah,1
        shl     ah,1
        shl     ah,1
        or      al,ah           ;Combine with low digit in AL
        ret

        public  _timeset
_timeset        proc    ;Set real-time clock to values in time structure
        push    bp              ;Save important registers
        mov     bp,sp
        push    si
        cld
        mov     si,4[bp]        ;Pointer to struct time argument
        lodsb                   ;Minutes in AL (hex)
        call    convert         ;Convert to BCD
        mov     cl,al
        lodsb                   ;Hours
        call    convert
        mov     ch,al
        inc     si              ;No hundredths
        lodsb                   ;Seconds
        call    convert
        mov     dh,al
        mov     dl,0            ;Set Daylight Savings Time flag off
        mov     ah,3            ;BIOS function number to set rtc time
        cli
        int     1ah             ;Set real time clock time
        sti
        pop     si
        pop     bp
        ret
_timeset        endp

        public  _dateset
_dateset        proc
        push    bp              ;Save important registers
        mov     bp,sp
        push    si
        cld
        mov     si,4[bp]        ;Pointer to struct date argument
        lodsw                   ;Year in AX
        mov     ch,20h          ;Assume century is 20,
        sub     ax,2000         ; calculate year in AL,
        jae     ok              ; and skip correction for century = 19
        mov     ch,19h          ;Correct for century = 19
        add     ax,100
ok:     call    convert         ;Convert year to bcd
        mov     cl,al
        lodsb                   ;Day
        call    convert
        mov     dl,al
        lodsb                   ;Month
        call    convert
        mov     dh,al
        mov     ah,5            ;BIOS function number to set rtc date
        cli
        int     1ah
        sti
        pop     si
        pop     bp
        ret
_dateset        endp
        end
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

LISTING 3

```
/*************************************************************************
*                    ZONCH.C - a time-zone change utility                *
*    ZONCH.COM (zone change) resets the real time clock to shift its     *
* time by an [integer] number of hours (positive or negative) entered    *
* on the calling command line.  ZONCH will not run if the zone change    *
* would cross midnight.                                                  *
*------------------------------------------------------------------------*
* Call with:  ZONCH <delta>   where <delta> is the number of hours to    *
*                             shift the real-time clock time.            *
*------------------------------------------------------------------------*
*                  Written by M. L. Lesser, 8/11/88                      *
*              Modified for Turbo C version 2.0, 10/29/88                *
*      Compiled with Turbo C (TCC) v 2.0, using options -mt -lt -O       *
*                and linked with TLINK v 2.0 to PCS.LIB                  *
*************************************************************************/

#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <private.h>    /* Contains function prototypes for PCS.LIB */

struct time now;
int    delta;

void main(int argc, char *argv[])
{

   if (argc == 1)
   {
      cprintf("\n\n\rUsage:  Call with ZONCH <delta> - where <delta> is"
                          " an integer number of "
              "\n\r       hours to shift the clocks.  Delta may be"
                          " positive, negative, or zero.\n\r");
      return;
   }
   delta = atoi(argv[1]);
   if (clktime(&now) != 0)
   {
      cputs("\nReal time clock inoperative.  Program terminated");
      return;
   }
   now.ti_hour += delta;
   if (now.ti_hour > 23)
   {
      cputs("\nTime change would have crossed midnight.  "
            "Program terminated."
          "\n\r     Clock times have not been changed.");
      return;
   }
   timeset(&now);               /* Set real-time clock */
   cprintf("\nThe corrected time is %02d:%02d:%02d\n\n\r",
           now.ti_hour,now.ti_min,now.ti_sec);
}
```

What's on the Menu?

by
Tom Swan

We must learn to explore all the options . . .
—James William Fulbright

Software menus are like swimsuits. Every year, it seems, a new fashion line appears, which programmers eagerly adopt in hopes of satisfying the public's desire for more daring pop-ups and more explicit pull-downs.

In past seasons, I've outfitted my code with just about every imaginable style of program-option menu. But I've only recently come to the conclusion that the typical menu procedure is about as utilitarian as a lead-filled lifesaver. Sometimes I swear if I have to dive into another case-statement-driven menu routine, I'd rather sink than swim.

Perhaps that's why the notion floated to mind recently that I'd never need to write a menu procedure again if my program's commands were able to present themselves for selection at run time. After all, what is a menu but a list of commands? So, why not let the commands organize themselves into their own menus?

That idea, which I'll explain in detail here, took shapely form while I was building the sand castles for an early prototype of a Turbo Pascal 5.5 object-oriented program, VA (for "View Anything"). I introduced VA, which will let me examine text, graphics, and other file types, in the previous issue of *PJ* (Vol. 7.6) and presented four of VA's low-level units: ukeys.pas, uitem.pas, ulist.pas, and ustritem.pas. This month, I'll complete the self-organizing menu system, which is usable as a stand-alone tool. Eventually, I plan to share all of VA's source code in *PJ* and to invite contributions from readers.

Taking Command

A "command," for our purposes, is an object that has a name and an activation character. For example, a command named "Open" might have the activation character O. In the simplest menu design, the activation character and keyboard key are the same, but I plan to add code to recognize function and control keys as well.

File ucmds.pas (Listing 1) defines the **command** object descended from **stritem** (explained last time). In typical object-oriented programming (OOP) fashion, the complete definition includes a related pointer and object data type:

```
commandPtr = ^command;
command = object( strItem )
    cmdCh : char;
    constructor init( theMenu: listPtr;
        ch : char; description : string );
    function getcmdCh : char; virtual;
end;
```

Type **commandPtr** defines a pointer to **command** object instances. The **command** object is an *abstract* type—it is never instantiated as a variable; rather, it serves only as a foundation for other object definitions (in this case, the actual program commands to come later). Interestingly, this new abstract type descends from the nonabstract **strItem**. Most OOP texts seem to suggest that abstract types are normally defined at the lowest level of an ancestral hierarchy, but as I learn more about OOP, I'm discovering plenty of room for new abstract types to fit in the middle of a family tree. Abstract types don't all have to be Adams and Eves.

Because **command** is descended from **strItem**, it inherits its ancestor's methods, constructors, destructors, and data fields. This means that **command** has a **cmdCh** field—the command's activation character—and also an **sp** string pointer field from its immediate ancestor **strItem**. Layering objects this way keeps descendant objects simple, but it can also hide important details from view. Good documentation is a must for keeping track of descendant genealogies.

Command also inherits and redefines its ancestor's **init** constructor. Three initialization parameters prepare program commands: **menu** of type **listPtr**, which addresses the list to hold the command; an activation character **ch**; and the command's name in a **description** string. The code for **init** is simple, performing only three statements:

```
cmdCh := ch;
strItem.init( description );
theMenu^.insertItem( @self );
```

After assigning the activation character to the object's **cmdCh** field, **description** is passed to the ancestor **init** constructor—a common OOP technique. Typically, descendant object constructors initialize the object's local fields (the single **cmdCh** in this case), and then continue the initialization process by calling the ancestor's constructor. After that step, **command** inserts itself into

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

theMenu list by passing its own address via **@self** to **insertItem**.

Soup du Jour

The next job is to design an object to hold menu items. Remember, a *menu* is defined here as a list of commands; therefore, it's natural that **menu** be descended from last month's **list** object. For reference, the object definition from umenu.pas (Listing 2) minus the method headers is repeated here:

```
menuPtr = ^menu;
menu = object( list )
   menuRow, menuCol : word;    { Menu command-line
                                 location }
   menuTitle : ^string;        { Menu name string }
   menuDisplay : ^string;      { Menu commands
                                 string }
   menuAttr : word;            { Display attribute }
{ ----- constructor, destructor, and other method
        headers }
end;
```

As usual, the complete definition has two parts. Pointer **menuPtr** addresses a **menu** object type, which contains five data fields in addition to those inherited from ancestors. In the present design, a **menu** has a location on screen (**menuRow** and **menuCol**), a name (**menuTitle**), a display string (**menuDisplay**), and a display attribute (**menuAttr**). The string fields are defined as pointers to conserve space that would otherwise be wasted by partially filled fixed-length string variables.

In keeping with standard OOP technique, a constructor **init** and destructor **done** handle the object's initialization and deinitialization duties. The constructor assigns values to the object's fields before completing the initialization by calling the **list** ancestor's **init** method. The destructor is a bit more complex than usual because it has to check whether the two string pointers are **nil** before disposing any memory allocated to the variables. Notice that after calling **dispose**, the pointers are set to **nil**, allowing the destructor to function properly even if the object is not completely initialized. This is important because the **init** constructor calls **done** if it runs out of memory at the call to **getMem** before any space is allocated for the **menuDisplay** pointer. In OOP, it's a good idea to test for **nil** pointers before passing them to **dispose**, especially inside a destructor's implementation.

Self-Optimizing Lists

There's not much worse than a sluggish option menu; so, to ensure top speed, I decided to have the **menu** object build the command-name string at run time. Doing that is the job of method **menuStr**, which extracts command names from the menu list and appends each name to a string addressed by **menuDisplay**. The inner **repeat** loop deserves close inspection:

```
repeat
   s2 := commandPtr( currentItem )^.getString;
   if length( s2 ) > 0
      then s1 := s1 + ' ' + s2;   { Add command
                                    to s1 }
   nextItem;
until atHeadOfList;
```

The loop adds a space plus each **command** name, obtained by calling method **getString**, to a temporary string **s1**. As you can see in the complete listing, after the loop finishes, **menuDisplay** addresses a copy of that string stored in the heap.

Building the display string this way improves runtime speed by minimizing the number of times the program has to examine the menu list. Calling **menuStr** returns the string that was constructed during a previous function call. If the menu changes at a later time, disposing of **menuDisplay**'s heap space and setting the pointer to **nil** triggers **menuStr** to rebuild a display string of new menu command names.

One negative aspect of this repeat loop is the unfortunate type cast required to convert **currentItem** to **commandPtr** for access to the inherited **getString** method. There seems to be no good way to avoid this kind of OOP bottleneck. Because **list** was defined earlier as a collection of linked **item** objects, a type cast is needed to process a list of **command** objects, which

I've only recently come to the conclusion that the typical menu procedure is about as utilitarian as a lead-filled lifesaver.

didn't exist when I wrote the ulist.pas module.

A few other **menu** methods—**getMenuAttr**, **getMenuRow**, and so on—have obvious purposes; I won't go into them in detail here.

Menus and Polymorphism

Four virtual methods in **menu** show off the power of polymorphism—the ability for later programming to affect the run-time action of precompiled code. The first of these methods, **displayMenu**, writes the current menu display string at **menuCol** and **menuRow**. Because **displayMenu** is a virtual method, a later object can replace it with more sophisticated programming; therefore, the menu system is not limited to its current single-line design.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The next two virtual methods, **beforeCommand** and **afterCommand**, do either nothing or very little. The methods serve as hooks into the menu display process, giving programs the opportunity to perform work before and after program commands are selected. These are examples of *abstract methods*, which are designed to be replaced by real code in descendants. Even though the methods aren't yet needed, they must be defined in the **menu** object now to allow the next virtual method to call them.

That method, **PerformCommands**, is guaranteed to end when receiving a kill signal—defined here as an <Esc> character. The method's inner repeat loop puts polymorphism to work:

```
repeat
   if ch = commandPtr( currentItem )^.getcmdCh then
   begin
      currentItem^.processItem;   { Perform command }
      resetList;                  { Force loop to
                                    end }
      afterCommand;               { Host post-cmd
                                    hook }
   end else
      nextItem;
until atHeadOfList;
```

After reading a keypress and storing the character in **ch**, the repeat loop begins with an **if** statement that compares **ch** to each command's activation character obtained by calling **getcmdCh**. Because that method is virtual, later code can replace it—perhaps to return a menu command ID number in the range 0–255 instead of an ASCII character. As in the **menuStr** loop, a type cast is necessary to remold **currentItem** into a **commandPtr** for access to **getcmdCh**.

Notice how **currentItem^.processItem** neatly performs the program's command. In fact, all program options are activated by that same statement, thanks to the magic of polymorphism. Eventually, the program will supply the command's actual programming by replacing the **item** object's virtual **processItem** method. Code written so far will not require modification to handle program options added in the future.

Before listing an example of how to create the program option modules, I should explain the final two virtual methods in **menu**. **InsertItem** and **removeItem** extend the methods of those same names as originally defined in the **list** object. The new methods call the originals after disposing any memory allocated to **menuDisplay**. This causes the next call to **menuStr** to rebuild the display string, which is necessary if any changes are made to the menu list. As a result, programs can insert, delete, and modify dynamic menus at run time.

Time Out

It occurs to me that some readers may be questioning, "Where is the menu's list pointer?" If that's bothering you, remember that **menu** *is* the list object—it doesn't *contain* a pointer to a list, as it probably would if I had defined it as a common Pascal **record**. This simple idea of objects performing work on themselves may be difficult to grasp initially, especially for experienced Pascal programmers who are used to procedural concepts where code performs work on data. Then again, perhaps I'm stating the obvious to an audience of believers. Anyway, *I* found the concept a bit tough to learn at first.

Building Extensible Programs

You may be surprised to discover that this completes the self-defining menu system. And there's not a single **case** statement in the lot! The remaining listings in this article demonstrate how to use the **command** and **menu** objects to build extensible programs.

File uprogram.pas (Listing 3) serves as the test program's main module. Storing main program actions in a unit allows for easier future modifications, as I'll demonstrate in a moment.

The unit's implementation defines the first program command—an object named **quitObj** and descended from **command:**

```
quitObj = object( command )
   procedure processItem; virtual;
end;
```

How simple that is! A **quitObj** is a **command** with a single virtual method, **processItem**, that performs the command's chores—in this case, whatever is needed to end the program. For test purposes, the method's implementation pushes an escape character into the ukeys.pas module to end the **performCommands** repeat loop, as explained earlier. A more complete example might also close files and carry out other shutdown duties.

The main unit's lone public procedure, **run**, initializes the **quitCmd** instance of type **quitObj** by calling the **init** constructor:

```
quitCmd.init( @theMenu, 'Q', 'Quit' );
```

Although **quitCmd** is a simple variable, it could be a pointer initialized by Turbo Pascal's extended **new** procedure. In later installments, using **new** to allocate space for object instances on the heap will be more important than it is now.

The rest of **run** calls **theMenu**'s **performCommands** method to activate the program. When you select the Quit command or press <Esc>, a few statements display a before-and-after **memavail** report to verify that all structures are properly disposed. You can activate that report by compiling the unit and defining a DEBUG symbol. To do that from DOS, run the command-line compiler by giving the following command:

```
tpc -dDEBUG uprogram
```

Uprogram's final statement initializes the self-defining menu by calling **theMenu**'s **init** constructor. This action occurs automatically at run time along with other unit initializations.

Finishing the Program

Two more units and a program finish the first example of the self-organizing menu system. This may seem to be an excessive number of files for a simple menu shell demo, but I want to show how extreme modularity leads to OOP code that's easy to extend with a minimum of

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

work. Down the road, that concept will play an essential role in VA's ability to learn how to read new file types.

Unit unew.pas (Listing 4) defines a global object, **newObj**, similar to **quitObj** in uprogram.pas. The command displays the message "You selected the New command" as proof that the code runs correctly. (Because this is only a test program, no commands perform any useful functions.) Most important is the initialization statement:

```
newCmd.init( @theMenu, 'N', 'New' );
```

The object's **init** constructor inserts the module's command into the menu, making unew.pas completely self-contained. No modifications to other modules are necessary to create new commands or to add their names to the program's menu.

Just to give the test program something else to do, a second module uhelp.pas (Listing 5) defines **helpObj**. Most of the details in this module are the same as in unew.pas.

And, finally, we come to the main program—test1.pas (Listing 6). The program uses only the main module unit **uprogram** plus the individual command modules **uhelp** and **unew**. The test program then calls **run** to start the show.

Modifying Menus

A simple experiment demonstrates how easy it is to customize a menu. Suppose you decide that you don't need the on-line help module in your code file. To take it out, just remove **uhelp** from **uses**:

```
uses uprogram, unew;
```

Recompile test1.pas and run. Everything works as before, but this time, the displayed menu lacks a help command. Suppose you later decide to reinsert the help module, and you also want to rearrange the commands in the menu. To do that, simply reorder the modules in **uses**:

```
uses uprogram, unew, uhelp;
```

Notice that test1.pas is the *only* source code required to make these modifications. Such flexibility allows you to design programs that people can customize without requiring access to your sensitive source text.

Multilevel Menus

The OOP menu system is also capable of handling multilevel command trees. To add a new menu level to the tree, write a new module with its own menu list object, as demonstrated in ufiles.pas (Listing 7).

The three new commands in ufiles.pas—**fListObj**, **fDirObj**, and **fEndObj**—are inserted into a private menu list object, **fileMenu**, a variable in the unit's implementation section. (The menu doesn't have to be a global variable; it could be declared dynamically on the heap, or it could be a local variable in a procedure or function.) The **fileObj** method **processItem** contains code that's very much like the main program's **run** procedure, which we saw earlier.

To link the new menu level to its parent, the unit's initialization code inserts a "File" command into the main program menu. When you press <F>, the module displays the new menu, giving access to the three new commands. The commands are activated by a recursive call to **menu.performCommands**, which ends when you press <Esc> or <Enter>. This type of subtle recursion, which is easy to miss when reading through source code, is commonplace in OOP. In fact, it's not unusual for many levels of recursion to exist among various virtual methods called by methods compiled earlier.

The final program, test2.pas (Listing 8), is identical to test1.pas except for the addition of **ufiles** to **uses**. The program is highly modular—you can strip out an entire menu level just by removing the unit name from **uses** and recompiling the small main program, which takes only a few seconds. Ideally, a separate linker would allow rebuilding menus by relinking the appropriate object-code files. Unfortunately, that's not possible with Turbo Pascal, which uses its own built-in linker.

Is It Practical?

It may be fair to question at this point whether 800 lines of code and two articles are overkill for what amounts to a rather simple menu shell. Has the skipper of this piece jumped overboard?

I don't think so, but I am finding that OOP code tends to be much longer than I had expected. Perhaps this is because I'm used to writing short, low-level modules before tackling more-hefty program jobs. Or maybe I'll discover that the extra-length, low-level OOP code inevitably leads to more-concise, higher levels. But I'll need to strengthen my OOP sea legs before I'll be able to answer that question.

Despite those few reservations, I'm pleased with the outcome. My extensible menu system lets me design program modules that are self-contained and can handle their own branches of a multilevel menu tree. During execution, only active menu elements need to be in RAM, and by converting selected modules to overlays, I can store code and associated menu data on disk until needed. I'm also finding Turbo Pascal's OOP extensions a pleasure to use. Most of the method implementations are as brief as bikinis. Customizing program menus is as easy as floating in a salt lake. And the modular units are as reusable as skin-tight nylon. So far, VA is swimming along nicely.

In coming installments, I'll uncover more of VA's midsections and have more fun in the sun while exposing OOP techniques for text, graphics, file handling, directory management, and other subjects. Feel free to send comments to me in care of *PJ*, or send an EasyPlex message to me via CompuServe (73627,3241). PJ

Tom Swan is a contributing editor of Programmer's Journal *and the author of* Mastering Turbo Pascal 5.5 *and* Mastering Turbo Assembler, *both published in 1989 by Howard W. Sams, Indianapolis, Indiana.*

Code follows.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disks.
ORDER BY PHONE — VISA/MC orders call toll free 800-234-0386.
Mail-order information on pages 8 and 88.

Code from "What's On the Menu?," by Tom Swan

1

LISTING 1

```
(* ucmds.pas -- (c) 1989 by Tom Swan *)

unit ucmds;

interface

uses crt, ustritem, ulist;

type

   commandPtr = ^command;
   command = object( strItem )
      cmdCh : char;
      constructor init( theMenu: listPtr; ch : char; description : string );
      function getcmdCh : char; virtual;
   end;

implementation

{ ----- Initialize a new command and insert into theMenu. }

constructor command.init(
   theMenu: listPtr; ch : char; description : string );
begin
   cmdCh := ch;
   strItem.init( description );
   theMenu^.insertItem( @self );
end;

{ ----- Return a command's character. }

function command.getcmdCh : char;
begin
   getcmdCh := cmdCh;
end;

end.
```

LISTING 2

```
(* umenu.pas -- (c) 1989 by Tom Swan *)

unit umenu;

interface

uses crt, ukeys, uitem, ucmds, ulist;

type

   menuPtr = ^menu;
   menu = object( list )
      menuRow, menuCol : word;    { Menu command-line location }
      menuTitle : ^string;        { Menu name string }
      menuDisplay : ^string;      { Menu commands string }
      menuAttr : word;            { Display attribute }

      constructor init( col, row : word; title : string; attr : word );
      destructor done; virtual;
      function menuStr : string;
      function getMenuAttr : Word;
      function getMenuRow : Word;
      function getMenuCol : Word;

      procedure displayMenu; virtual;
```

2

```
      procedure beforeCommand; virtual;
      procedure afterCommand; virtual;
      procedure performCommands; virtual;

{ ----- Replacement methods inherited from list object. }

      procedure insertItem( ip : ItemPtr ); virtual;
      procedure removeItem( ip : ItemPtr ); virtual;

   end;

implementation

const

   ESC = #27;      { ASCII escape char }

{ ----- Initialize a new menu command list. }

constructor menu.init( col, row : word; title : string; attr : word );
begin
   menuRow := row;
   menuCol := col;
   menuAttr := attr;
   menuDisplay := nil;  { Created by menu.menuStr }
   getMem( menuTitle, length( title ) + 1 );
   if menuTitle = nil then
   begin
      fail;
      done;
   end else
      menuTitle^ := title;
   list.init;
end;

{ ----- Dispose of a menu command list. }

destructor menu.done;
begin
   if menuTitle <> nil then
   begin
      freeMem( menuTitle, length( menuTitle^ ) + 1 );
      menuTitle := nil;
   end;
   if menuDisplay <> nil then
   begin
      freeMem( menuDisplay, length( menuDisplay^ ) + 1 );
      menuDisplay := nil;
   end;
   list.done;
end;

{ ----- Create and/or return current menu string. }

function menu.menuStr : string;
var
   s1, s2 : string[80];
begin
   if menuDisplay = nil then
   begin
      s1 := '';              { Null string }
      if not listEmpty then
      begin
         resetList;
         repeat
            s2 := commandPtr( currentItem )^.getstring;
            if length( s2 ) > 0
               then s1 := s1 + ' ' + s2;  { Add command to s1 }
            nextItem;
         until atHeadOfList;
      end;
      getMem( menuDisplay, length( s1 ) + 1 );
      if menuDisplay <> nil
         then menuDisplay^ := s1;
   end else
```

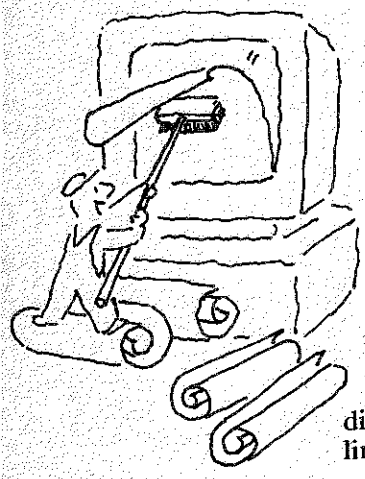
Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Two great tools.

NEW VERSION 2.8
CLOCK & EDIT UNITS, DIALOG BOXES, AND PROGRESS BARS!

SAYWHAT?!
The lightning-fast screen generator.



Whether you're a novice programmer longing for simplicity, or a seasoned pro searching for higher productivity, you owe it to yourself to check out Saywhat. You see, with Saywhat, you can build beautiful, elaborate, color-coded screens in minutes! That's right. Truly *fantastic* screens for menus, data entry, data display, and help-panels that can be displayed with as little as one line of code in *any* language.

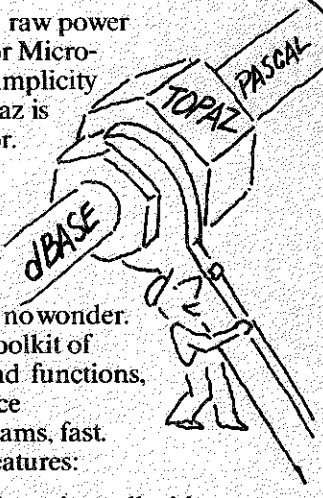
Here's what you get:

- Design screens, windows, and *moving bar menus!*
- Easy-to-use, powerful editor lets you create screens in a jiffy.
- Pop up your screens and menus with one line of code in dBASE, all the dBASE compilers, your favorite BASIC, Pascal, *or any other language!*
- Screen Library Manager.
- Generates runtime code.
- No runtime license or royalty fees.
- Comes with a 100 page manual, plus dozens of sample programs and free utilities.

$49.95

TOPAZ
The breakthrough DBMS toolkit for Pascal

If you'd like to combine the raw power and speed of Turbo Pascal or Microsoft's QuickPascal with the simplicity and elegance of dBASE, Topaz is just what you're looking for. That's because Topaz was specially created to let you enjoy the best of *both* worlds. The result? You create complete, truly dazzling applications in a very short time. And no wonder. Topaz is a comprehensive toolkit of dBASE-like commands and functions, designed to help you produce outstanding, polished programs, fast. Check out these powerful features:



- Over 220 routines all with easy-to-use, dBASE-like syntax.
- Data entry routines like **SAY, GET, PICTURE, RANGE,** color selection, unlimited data validation.
- Open up to 10 DBF files, with up to 15 indexes with **USE, SELECT, SKIP, APPEND, PACK, INDEX ON, SET INDEX TO,** and **FIND.**
- No need to buy dBASE. **CREATE, BROWSE** and **REPORT** utilities included.
- Easily implement Saywhat and Lotus-style moving bar menus.
- **BROWSE** or **EDIT** any DBF file with just one line of code! Programmable and windowed too.
- Pick from windowed data or filenames with one line of code.
- Comprehensive Time & Date math in 7 international formats.
- Powerful code and report generators included!
- Comes with a complete 320 page manual, plus sample programs to get you started.

$74.95

ORDER NOW. YOU RISK NOTHING.

Thousands of satisfied customers have already ordered from us. Why not visit your dealer or call toll-free, right now and put Saywhat and Topaz to the test yourself? They're fully guaranteed. You don't risk a penny.

Special limited-time offer! Save $26. Buy Saywhat?! and Topaz together for just $99 (plus $5 shipping and handling, $10 Canada, $25 International, Calif. residents add 7%).

Visit your nearest dealer
or call toll-free:
800-468-9273
In California: 800-231-7849
International: 415-571-5019

Software Science, Inc.
100 Valley Drive, Brisbane, CA 94005

MONEY BACK GUARANTEE.
If you aren't completely delighted with Saywhat or Topaz, for any reason, return them within 30 days for a prompt, friendly refund.

Dealers: SAYWHAT?! and TOPAZ are available from Kenfil Distribution, and in Europe from ComFood Software, W. Germany 49-2534-7093

Guaranteed!

SOFTWARE SCIENCE INC

Number 198 on the Reader Service Card

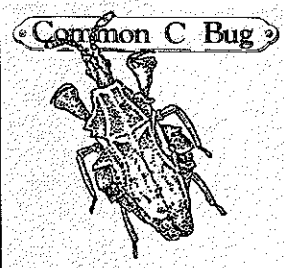
PROVIDED BY WWW.8BIT.ORG
ELECTRONIC REPRODUCTION PROHIBITED

T-SHIRTS

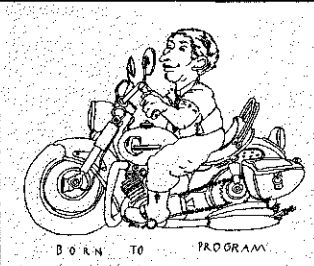
Multi-color imprints on the highest quality 50/50 cotton T-shirts.

$12.00 each – includes shipping & handling

SIZES S, M, L, XL

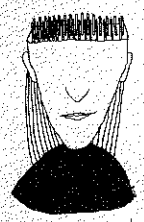


COMMON C BUG



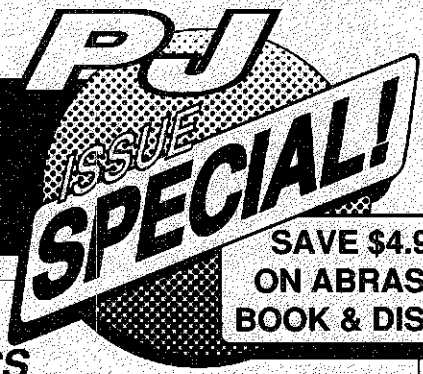
BORN TO PROGRAM

PARALLEL PORTS



RS-232 HEAD

BUGS RULE 4 color imprint on a white shirt



SAVE $4.90 ON ABRASH BOOK & DISK

POWER GRAPHICS PROGRAMMING

ABRASH BOOK AND CODE DISK.

For a limited time, you can save over 10% when you order both the *Power Graphics Programming* book and the companion code disk during the ***PJ*** 8.1 Issue Special. Order your book and disk today for only $35.00 and save $4.90.
(Offer expires February 28, 1990. All orders must be prepaid.)

BACK ISSUES

$7.00 each PREPAID ONLY

A few of the Back Issues currently available

Vol. 6.2: Hal Hardenbergh on RISCs, CISCs, and Fabs; Pink Noise; yet another VGA write mode; inside Turbo Pascal 4.0; the proposed ANSI C language standard; converting a Turbo C program to a TSR; 80286 and 80386 programming; 80386 software development; *The Norton Guide for BIOS/DOS Assembly.*

Vol. 6.3: Hal Hardenbergh on memory trends; Pink Noise; getting your consulting tools in order; Gordon Letwin on OS/2 memory management; virtualizing the 386; Pascal—a CASE of the jumps; Turbo Professional 4.0; BASIC—CALL, GOSUB, and multiline functions.

Vol. 6.4: Job shopping; windowing systems overview; reading VGA memory; X Window—a view from DOS; a digital clock in C for MS Windows; windowing in TopView, DESQview, and MS Windows; MS C 5.1 and the ANSI standard; a mouse unit for Turbo Pascal 4.0.

Vol. 6.5: Hal Hardenbergh on computers in the fast lane; Consulting—no-win jobs and too much work; VGA text paging; using dynamic arrays and records in Turbo Pascal 4.0; an OS/2 Presentation Manager application template; WATCOM C 6.0 and the ANSI standard; object-oriented OS/2 in C++; Ada and software engineering; REXX on the PC.

Vol. 6.6: PC DOS 4.0; OS/2 vs. UNIX; VGA screen saving and color selection; MASM quirks; addressing extra memory for DOS; DOS device drivers debugging and information utility; proposed ANSI C language standard; Ray Duncan on Forth; Ada data types and exceptions; PC floating-point hardware and software.

Vol. 7.1: GRAPHICS: High-resolution VGA color modes; graphics processors; modifying type constants stored in Turbo Pascal 4.0 and 5.0 compiled EXE code; comparing Pascal overlay managers; JPI's TopSpeed Modula-2; faster QuickBASIC code; Microsoft's CodeView debugger; Lattice C V3.3; smart terminal emulations in Windows; the COBOL-85 ANSI standard; Pink Noise—industry news and reviews.

Vol. 7.2: OPERATING SYSTEMS: undocumented DOS features; fast 80386 block moves under DOS; OS/2 dynamic link libraries; OS/2 multitasking; debugging utility for PM and Windows; EGA and VGA animation; disk formatting in Pascal; how to become a reseller; reviews of windowing and graphics tools, disassemblers, and *The 80386 Book.*

Vol. 7.4: COMMUNICATIONS: "Pink Noise," by Frank Greco; Michael Abrash's "On Graphics"; a simple terminal program in PM; communications and protocols; better coding conventions for assembly language; implementing the Boyer–Moore search algorithm in assembly language; database normalization theory and practice; Turbo Pascal 5.5 development tools; Lint utilities for the PC; a millisecond timer for BASIC; exit codes for QuickBASIC; reviews of RamNet, DOS/16M, PCX *Technical Reference Manual* and Programmer's Toolkit, GRASP 3.1, AXOS, and *Microsoft Mouse Programmer's Reference.*

Vol. 7.5: OBJECT-ORIENTED PROGRAMMING: Class libraries in C++; from C to objects; improved C++ code generation; Hal Hardenbergh on the RISC/CISC schism; handling version management on large projects; "Still-Higher 256-Color VGA Resolutions," by Michael Abrash; a library system for assembly language; adding world scaling to Turbo Pascal's BGI graphics primitives; numeric extensions to C; reviews of 386 development environments and tools, mini-case tools, and Robert Sedgewick's book *Algorithms.*

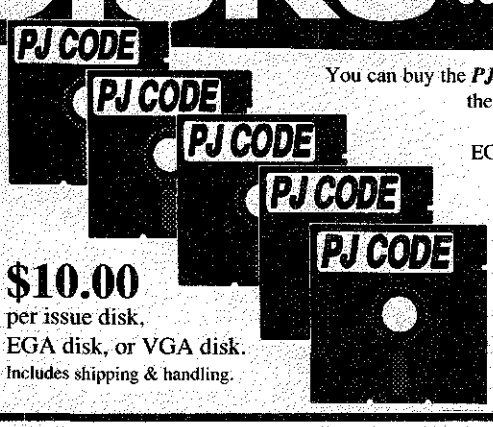
Vol. 7.6: USER INTERFACES: Windows terminal program; the CURSES C library; using two video displays with Turbo Pascal; Michael Abrash on VGA and EGA split screens; memory models and static variables for assembly language; ANSI C standard; Aztec C version 4.2; using object-oriented programming to create a file viewer; accessing large memory spaces from DOS on the 80386; making use of QuickBASIC's string-searching algorithm; reviews of user-interface tools and mini-Case tools; the *System BIOS for IBM PC/XT/AT* series, by Phoenix Technologies.

10% discount on any order of 10 or more T-Shirts, Disks or Back Issues.

All mail orders must be accompanied by check or Money Order -- PREPAID ONLY.

CODE DISKS

Use the order form on the card opposite.
Order by phone with your VISA/MC.
CALL TOLL FREE
800/234-0386 ext 53



$10.00
per issue disk,
EGA disk, or VGA disk.
Includes shipping & handling.

You can buy the ***PJ*** code listings from volume 5.5 to the current issue. Code disks for the Michael Abrash "On Graphics" EGA/VGA series are also available.

Diskettes are available for the following issues:

5.5	6.3	7.1	7.5	VGA
5.6	6.4	7.2	7.6	
6.1	6.5	7.3	8.1	
6.2	6.6	7.4	EGA	

360K MS-DOS disks standard; other formats available.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Swan code, continued

3

```
      s1 := menuDisplay^;
   if menuTitle <> nil
      then s1 := menuTitle^ + s1;
   menuStr := s1;
end;

{ ----- Return menu attribute word. }

function menu.getMenuAttr : Word;
begin
   getMenuAttr := menuAttr;
end;

{ ----- Return menu row. }

function menu.getMenuRow : Word;
begin
   getMenuRow := menuRow;
end;

{ ----- Return menu column. }

function menu.getMenuCol : Word;
begin
   getMenuCol := menuCol;
end;

{ ----- Display the menu name and command strings. }

procedure menu.displayMenu;
var
   oldAttr : word;    { For saving current attribute }
begin
   oldAttr := textAttr;
   textAttr := menuAttr;
   gotoxy( menuCol, menuRow );
   write( menuStr );
   clreol;
   textAttr := oldAttr;
end;

{ ----- Called at the top of the menu.performCommands repeat loop,
providing host programs a way to hook into the keyboard polling loop.
Optionally replaced by host program's menu object. }

procedure menu.beforeCommand;
begin
end;

{ ----- Called after menu.duringCommand. Optionally replaced by host
program's menu object. Note: you may use this procedure to modify the
menu list, allowing one command to alter the availability of other
commands. }

procedure menu.afterCommand;
begin
   displayMenu;    { Usually a good idea }
end;

{ ----- Process commands in menu list. Guaranteed to return upon
pressing <Esc>. Repeatedly calls beforeCommand while polling keyboard
for input. Calls afterCommand after processing a selected command in
the menu list. }

procedure menu.performCommands;
var
   ch : char;
begin
   displayMenu;          { Display menu name and commands }
   ch := chr( 0 );       { Initialize ch to null }
   repeat
      beforeCommand;        { Activate host polling hook }
      if keyWaiting then
      begin
         ch := upcase( getKey );
```

4

```
         if ch <> ESC then
         begin
            resetList;
            if not listEmpty then
            repeat
               if ch = commandPtr( currentItem )^.getcmdCh then
               begin
                  currentItem^.processItem;  { Perform command }
                  resetList;                 { Force loop to end }
                  afterCommand;              { Host post-cmd hook }
               end else
                  nextItem;
            until atHeadOfList;
         end;
      end;
   until ch = ESC;       { Until <Esc> pressed or forced }
end;

{ ----- Insert a new command into menu. Parameter ip must address a
command object as defined in ucmds.pas. Also dispose of the current
menu display string if neccessary, as these are invalid after new
commands are added to the list. }

procedure menu.insertItem( ip : ItemPtr );
begin
   if menuDisplay <> nil then
   begin
      freeMem( menuDisplay, length( menuDisplay^ ) + 1 );
      menuDisplay := nil;
   end;
   list.insertItem( ip );
end;

{ ----- Delete a single command from the menu.  Parameter ip should
address a command object (or a descendant). Also dispose the current
menu display string. }

procedure menu.removeItem( ip : ItemPtr );
begin
   if menuDisplay <> nil then
   begin
      freeMem( menuDisplay, length( menuDisplay^ ) + 1 );
      menuDisplay := nil;
   end;
   list.removeItem( ip );
end;

end.
```

LISTING 3

```
(* uprogram.pas -- (c) 1989 by Tom Swan *)

unit uprogram;

interface

uses crt, ukeys, ucmds, ulist, umenu;

procedure run;

var

   theMenu  : menu;

implementation

{ ----- Private command object. }

type

   quitObj = object( command )
```

Code continues . . .

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Swan code, continued

5

```
      procedure processItem; virtual;
   end;

var

   quitCmd  : quitObj;

{$IFDEF DEBUG}
   startmem : longint;  { Holds memavail at startup for debugging }
{$ENDIF}

{ ----- Run the program. }

procedure run;
begin
   quitCmd.init( @theMenu, 'Q', 'Quit' );
   theMenu.performCommands;
   theMenu.done;
   gotoxy( 1, hi( windmax ) + 1 );
   writeln;
{$IFDEF DEBUG}
   writeln( 'mem at start = ', startmem );   { display statistics }
   writeln( 'mem at end   = ', memavail );
{$ENDIF}
end;

{ ----- Implement the Quit command. }

procedure quitObj.processItem;
begin
   pushKey( chr(27) );      { Simulate <Esc> keypress }
end;

begin
{$IFDEF DEBUG}
   startmem := memavail;   { Record memavail at startup }
{$ENDIF}
   theMenu.init( 1, hi( windmax ) + 1, 'Test:', WHITE + BLUE * 16 );
end.
```

LISTING 4

```
(* unew.pas -- (c) 1989 by Tom Swan *)

unit unew;

interface

uses crt, ucmds, uprogram;

type

   newObj = object( command )
      procedure processItem; virtual;
   end;

var

   newCmd   : newObj;

implementation

procedure newObj.processItem;
begin
   clrscr;
   writeln( 'You selected the New command' );
end;

begin
   newCmd.init(  @theMenu, 'N', 'New'  );
end.
```

6

LISTING 5

```
(* uhelp.pas -- (c) 1989 by Tom Swan *)

unit uhelp;

interface

uses crt, ucmds, uprogram;

type

   helpObj = object( command )
      procedure processItem; virtual;
   end;

var

   helpCmd  : helpObj;

implementation

{ ----- Display sample help screen. }

procedure helpObj.processItem;
begin
   clrscr;
   writeln( 'Help!' );
   writeln;
   writeln( 'Type the first letter of a command.  Type Q' );
   writeln( 'or <Esc> to quit and return to DOS.' );
end;

begin
   helpCmd.init( @theMenu, 'H', 'Help' );
end.
```

LISTING 6

```
(* test1.pas -- (c) 1989 by Tom Swan *)

program test1;
uses uprogram, uhelp, unew;
begin
   run;
end.
```

LISTING 7

```
(* ufiles.pas -- (c) 1989 by Tom Swan. *)

unit ufiles;

interface

uses crt, ukeys, ucmds, umenu, uprogram;

type

   fileObj = object( command )
      procedure processItem; virtual;
   end;

var

   fileCmd  : fileObj;
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
implementation

type
    fListObj = object( command )
      procedure processItem; virtual;
    end;

    fDirObj = object( command )
      procedure processItem; virtual;
    end;

    fEndObj = object( command )
      procedure processItem; virtual;
    end;

var
    fListCmd : fListObj;
    fDirCmd  : fDirObj;
    fEndCmd  : fEndObj;
    fileMenu : menu;

{ ----- File-menu methods. }

procedure fListObj.processItem;
begin
    clrscr;
    writeln( 'File-List command' );
end;

procedure fDirObj.processItem;
begin
    clrscr;
    writeln( 'File-Dir command' );
end;

procedure fEndObj.processItem;
begin
    pushKey( chr(27) );  { Ends file menu processing }
end;

{ ----- Run the file menu. }

procedure fileObj.processItem;
begin
    clrscr;
    fileMenu.init( 1, hi( windmax ) + 1, 'File:', WHITE + BLUE * 16 );
    fListCmd.init( @fileMenu, 'L', 'List' );
    fDirCmd.init(  @fileMenu, 'D', 'Dir' );
    fEndCmd.init(  @fileMenu, chr(13), '' );  { <Enter> ends menu }
    fileMenu.performCommands;
    fileMenu.done;
end;

begin
    fileCmd.init( @theMenu, 'F', 'File' );
end.
```

LISTING 8

```
(* test2.pas -- (c) 1989 by Tom Swan *)

program test2;
uses uprogram, ufiles, uhelp, unew;
begin
    run;
end.
```


TechnoJock's Turbo Toolkit

Attention Turbo Pascal programmers.....

Cut your program development time in half and create sophisticated user interfaces in minutes with *TechnoJock's Turbo Toolkit!*

The *Toolkit* contains twelve units with all source code included, and a 300 page manual full of examples. These range from a comprehensive unit that provides a Sidekick Plus style directory window with all the expected sorting features, to a basic procedure that writes text centered on the screen.

And it's easy to use! If you've bought other toolkits but didn't use them because of all their complicated rules ... TTT is for you.

TECHNOJOCK SOFTWARE

- **ultra fast screen writing**
- **EGA/VGA extended lines**
- **exploding boxes**
- **virtual screens**
- **sorted directory listings**
- **comprehensive form input**
- **pulldown menus**
- **nested menus**
- **string manipulation**
- **windows**
- **full mouse support**
- **keyboard control**
- **date manipulation**

"TechnoJock's Turbo Toolkit may just cut your work in half! It provides easy-to-implement procedures.... designed for all Turbo Pascal programmers. With this Toolkit everyone becomes a TechnoJock."

Turbo Users Group

"Here, despite the whimsical name, is shareware at its best: an innovative product, a superb manual, quick mail-order service, and a reasonable price... If you're programming in Turbo Pascal, you absolutely must have TTT!"

Public-Domain Software & Shareware 2nd edition

"Looks great! We have decided to implement your Toolkit in in-house programs for Tech Support."

Rich Kubota
Technical Support
Borland International

TechnoJock Software, Inc.
PO Box 820927
Houston, TX 77282-0927

Orders: 1-800-242-4775
Enquiries: (713) 493-6354

Only $49.95!

Plus $5.00 shipping and handling (Canada $10.00, overseas $15.00). Texas residents add 8% Sales Tax.

Send checks to TechnoJock Software. Send Visa/Mastercard details to: Public (Software) Library PO Box 35705 Houston TX 77232-5705

Number **289** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Six easy ways to boost your BASIC

PROBAS™ Basic Programming Library

So who cares that BYTE magazine calls **PROBAS** a "Supercharger for QuickBASIC" or that PC Tech Journal says that **PROBAS** is a "high-quality, high-quantity package"? Who buys a product just because Jerry Pournelle said "Anyone doing serious QuickBASIC programming would do well to get [**PROBAS**]"? And who cares that Wayne Hammerly calls **PROBAS** "The greatest thing since sliced bread"?

Who?--Only those who want to write better, faster, slicker programs and save hundreds of programming hours in the process. With all of that hoopla out of the way, we are formally announcing the momentous release of **PROBAS** Version 3.1, now with over 400 assembly routines to make BASIC programs faster and more powerful than you ever dreamed with features like:

- A 1,000-page two-volume manual
- Full mouse support
- Extended and EMS memory support
- Full-featured windowing
- Moveable, resizable windows
- Screen snapshots (text & graphics)
- Virtual screens in memory
- Lightning-fast file I/O
- Critical error handling
- String, array, and pointer sorts
- Search directories and archives

Create dazzling screens in text, CGA, EGA, VGA, and Hercules graphics modes with windows that can overlay one another and be moved and resized on the fly. Store megabytes of string, data, or screen snapshots in extended or EMS memory. Draw complex text or graphic screens to memory and snap them on in an eyeblink. The **PROBAS** file I/O routines allow you to read or write huge chunks of data at a clip, far faster than with BASIC.

PROBAS also has over 300 other essential services, including handy string, date, time, directory, and array manipulation routines; string, screen, and data compression routines; valuable equipment and input routines; and faster replacements for many BASIC commands.

Whether you are a professional or a novice, **PROBAS** will boost your BASIC in ways you never thought possible. **PROBAS** allows the professional to write faster, tighter code in much less time and allows novices to quickly and easily write professional-quality programs that would be impossible with BASIC alone. The bottom line is **PROBAS** adds power and saves time. After all, how much is a few hundred hours of your time really worth?

For all DOS versions of QuickBASIC and BASCOM. ***Just $149.00!***

Number 249 on the Reader Service Card.

PROREF™ On-Line Help For PROBAS

PROREF provides on-line help for the routines in the **PROBAS** library. This hypertext manual links directly to the QB Advisor in QuickBASIC 4.5 so that the **PROBAS** reference becomes an integral part of your QuickBASIC on-line manual. Includes information and examples on **PROBAS** routines and helpful hints on programming in BASIC. ***Just $50.00!***

PROSCREEN™ Screen Management

PROSCREEN is a full-featured screen generator/editor that will save you more design and coding time than you ever thought possible. **PROSCREEN** treats screens like a word processor treats text to provide complete control over characters, colors, and placement. Design input screens with up to 130 fields, 19 predefined and 2 user-defined masks. Save screens to screen files or .OBJ files and use the tight BASIC/Assembly code that comes with **PROSCREEN** to access the screens. There's no kludgy code generator here! Access hundreds of input screens with less than 25k of total code. ***Just $99.00!***

PROMATH™ Mathematics Library

PROMATH is a collection of over 150 high-level routines that provide mathematical functions and operations for programmers who often work in mathematics, science, or engineering. Complex variables, real and complex matrices, real and complex trigonometric and hyperbolic functions and their inverses, solution of linear equations, integration, differential equations, Fast Fourier transforms, graphics support, and many other useful routines are provided.

For years Fortran has been the language of choice for scientific and engineering applications, but it lacks many of the useful features of QuickBASIC. **PROMATH** contains most of the Fortran mathematical and numeric functions and allows you to easily translate Fortran code to BASIC or write new programs in BASIC, while retaining Fortran's numerical prowess.

The **PROMATH** manual is over 200 pages and provides a complete description of each routine, including any algorithm and the mathematical formula the routine uses, shown in standard notation. For QuickBASIC 4 and BASCOM 6 only. ***Just $99.00!***

PROBAS™ TOOLKIT

The **TOOLKIT** is a collection of high-level BASIC and assembly modules that use the routines in the **PROBAS** library to save you even more hours of grunt work. Why spend hundreds of hours re-inventing the wheel when you can just plug in **TOOLKIT** modules like:

- Super-fast B-Tree indexing
- Ring, Bar, Pop-Up, Pull-Down menus
- Scroll-bar tag windows
- Dialog boxes with radio buttons
- Two mini-editors with word wrap
- BCD math routines
- Julian date & calendar routines
- Patch .EXE files
- Protected memory storage area

The **TOOLKIT** now supports EGA and VGA graphics modes for menus, windows, editors, calendars, and more. Complete with BASIC source code and an all-new comprehensive manual. The **TOOLKIT** requires the **PROBAS** library and helps conserve your greatest asset of all—time! ***Just $99.00!***

PROBAS™ TELECOMM TOOLKIT

The **PROBAS TELECOMM TOOLKIT** is a collection of high-level communications modules that you plug into your code to provide popular file transfer protocols, terminal emulations, login scripts and baud rates up to 115,200 baud. You get:

- Xmodem/Modem7/Xmodem-1k
- Ymodem (single and batch)
- CRC-16 and Checksum
- VT52, VT100, ANSI, BBS etc.
- Auto Dialer & data base
- Documented BASIC source

Why use clumsy SHELLs to complex terminal programs when you can plug just the communications routines you need into your code? Implement just the features and commands you want. Requires **PROBAS**. ***Just $75.00!***

Our thirty-day, money-back guarantee assures you the highest quality and our technical support staff is always ready to help.

HAMMERLY COMPUTER SERVICES, INC.

9309 JASMINE COURT • LAUREL, MD 20707

(800) 343-7484

INT'L ORDERS: (301) 953-2191 FAX: (301) 725-8147
BBS: (301) 953-7738

Add $5.00 per item ($8.00 Canada) for shipping per order. Europe: $39.00 for 1st item plus $5.00 for each additional item. Visa, M/C, C.O.D. (US Only) checks and approved POs accepted. Trademarks **PROBAS**, **PROREF**, **PROSCREEN**, **PROMATH**, Hammerly Computer Services, Inc., QuickBASIC, BASCOM: Microsoft Corp.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Continued from page 10

No Hang-Ups, Thank You

I have finally gotten around to reading the letters in *PJ* 7.4 and found in [Steven Baker's] reply to the letter of George Dinwiddie a point that used to bother me also: "If a casual user invokes a program that expects input from stdin, they are provided with a system that looks like it's dead."

True enough most of the time. But it doesn't have to be. When I bought Turbo C Version 1.0, the linker could not provide a map file with alphabetized publics—at least I never figured how to persuade it to do so. So I tried to use the DOS sort filter to do this. It was so slow, I thought the system had hung, and I decided to write a quick replacement. I also wanted to deal with the problem [Baker] mentioned. I think you may need DOS 3.x or higher for the code to work, but it seems to solve the problem of the belly-up console. Here is the code:

```
DEVICE_INFO    RECORD
RES:1,CTRL:1,RESERVED:6,ISDEV:1,
                                _EOF:1,
BINARY:1
MASK    ISDEV = 80H
VER_OK:
;-------------------
;BE SURE THAT STDIN IS CONNECTED TO A
FILE, NOT A DEVICE.
;-------------------
        mov bx,STDIN
        mov ax,44H SHL 8
;AH = 44H IOCTL FOR DEVICES
        int 21h
;AL = 0 GET DEVICE INFO

        test dx,MASK ISDEV
        jz NOT_DEVICE

        mov dx,offset [DS:SYNTAX_MSG]
        mov cx,SYNTAX_MSG$
        jmps ABORT_EXIT
NOT_DEVICE:
;-------------
;parse command line
;-------------
```

Larry Smith
Friedland, West Germany

Re Recursive Reentrancy

In his article, "Better C++ Code Generation" (*PJ* 7.5), John M. Dlugosz incorrectly states that functions that return a structure in Zortech C++ are fully reentrant. They are reentrant, but only in a limited sense. If the function is only ever called from a single execution thread, the function is indeed reentrant. However, if the function is called from two or more threads that may execute asynchronously, then reentrance is sacrificed. To use John's own example, consider the function **avalue = foo()**, where **avalue** is a structure. If **foo()** were to be interrupted and suspended, say by a hardware interrupt handler or a preemptive multitasker, and **foo()** called during the interruption (sharing the same data space), then there is a possibility that the static buffer **foo()** uses to return its structure will be overwritten, depending upon precisely when the interruption occurs.

If the interruption occurs after the return data is copied (that is, after the **rep movsw**) and before it returns, then the second **foo()** invocation will copy its return data over top of the first invocation's return data. When the interrupted **foo()** is eventually resumed, it will incorrectly return the data left there by the second **foo()** invocation.

Mr. Dlugosz has confused recursive reentrancy with full reentrancy. He is not alone. Many in the microcomputer world, addicted to the single-tasking nature of DOS, very often miss distinctions such as this. The question to ask when considering reentrancy is not "Can this function call itself?" but "Can this function be interrupted at any point and survive a reinvocation of itself?"

Steve Barber
New York, New York

We at *Programmer's Journal* like to hear from our readers. Please address correspondence to Managing Editor, *Programmer's Journal*, 150 N. 4th St., Springfield, OR 97477. You can also contact us via fax (503-746-0071), CompuServe (user ID 76676,427), or MCI Mail (user ID 336-4248). All letters become the property of *Programmer's Journal* and may be edited for publication.

Unleash the BASIC Power of Hypertext!

Our latest BASIC-booster includes the first hypertext engine designed to be called from BASIC. With the **ProBas HyperHelp ToolKit** you can use the hypertext engine to put one or more manuals on-line with full hypertext search facilities. Imagine having one or more manuals on-line with all of the fantastic hypertext abilities in the QuickBASIC 4.5 compiler and then some:

- Choose single- or multi-window display
- Specify window colors and placement
- Move and resize windows with mouse or keys
- Sophisticated mouse and keyboard interface
- Up to 40 bookmarks to move between hyperlinks

In less than a dozen lines of code you can pop-up context-sensitive help at any time. Your users can then jump to related help, examples, or just browse the manual(s) in one or more windows that they can move or resize at will. The text will automatically wrap within the window to stay fully visible!

Create HyperCard Applications

Use the **HyperHelp** engine to create full-blown hypercard applications. Create sophisticated multi-window, multi-thread hypercard stacks. Mix cards and manuals for total data integration. Moving from link-to-link or card-to-card is instantaneous and the speed will amaze you.

Adding hypertext to your applications takes less than about a dozen lines of BASIC code. Converting ASCII text to hypertext is just as easy—just put delimiters around keywords, hyperlinks, and items you want to display in boldface.

Multiple Help Subsystems

The hypertext engine is just a part of the **HyperHelp ToolKit**. There is a wide selection of help subsystems with various displays, user interfaces, and memory requirements to suit almost any need. Want a small, window-oriented, keyword help system? It's in there! Need a Terminate & Stay Resident help system? It's in there! How about a lightbar indexing system that then pops-up the selected text? It's in there!

The **HyperHelp ToolKit** gets it's blinding speed by using the low-level routine in our **ProBas** Professional Basic Programming Library. **HyperHelp** requires the **ProBas** Library. See our ad on the opposite page for information on **ProBas**, shipping rates, and our thirty-day money-back guarantee.

Special Introductory Price—For a short time this powerful hypertext engine and collection of help subsystems is available for ***just $99.00!***

HAMMERLY
COMPUTER SERVICES, INC.
9309 JASMINE COURT • LAUREL, MD 20707
(800) 343-7484
INT'L. ORDERS: (301) 953-2191 FAX: (301) 725-8147
BBS: (301) 953-7738

Add $5.00 per item ($8.00 Canada) for shipping per order. Europe: $39.00 for 1st item plus $5.00 for each additional item. Visa, M/C, C.O.D. (US Only) checks and approved POs accepted. Trademarks ProBas, ProRep, ProScreen ProMath: Hammerly Computer Services, Inc. QuickBASIC, BASCOM: Microsoft Corp.

Number 293 on the Reader Service Card

COMING ISSUE—

March/April 1990 Volume 8.2

Graphics:
TIGA-340 Programming
by Jake Richter
Even Faster VGA Circle Drawing
by Michael Abrash
X Window under DOS
by Steven Baker

386 NOW:
Using Extended Memory Directly
by Larry Smith

Windows:
Dynamic Data Exchange
by William S. Hall

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Offer your product directly to Professional Programmers:

Placing an ad in **PJ BILLBOARD** is an easy, low-cost way to reach potential buyers. Regular listings include a headline, company information, and approximately 400 characters for your message. Display ads are 1 5/8" wide by 2 3/8" high. For more information or to reserve a space, call our ad department at **503-747-0800.**

EVERLOCK

COPY PROTECTION

Designed for clone compatibility & strength

- Thwarts ALL Bit-copy Software
- Protect any COM/EXE w/o Source changes
- Shut down Debug Tracing & Disassemblers
- Supports all Hard & floppy Disk formats
- Restrict or allow Network Installs
- Limit program use by Date or #Executes
- Remotely reset Installs, Date or #Executes
- No damaged media or I/O plugs

For IBM & clones--$195 & up--Free Info & Demo

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800) 227-0644 (816) 776-2700
FAX (816) 776-8398

Number 51 on the Reader Service Card

EVERTRAK

SECURITY

If you HATE Copy Protection, but still want to protect your program from hackers and reverse engineering, **EVERTRAK** allows you to:

- Protect any COM/EXE w/o Source changes
- Shut down DEBUG systems and disassemblers
- Embed/display hacker-proof Customer information
- Restrict use by date-range and/or media-type
- Track unauthorized copies back to their source
- Totally invisible to End Users

EVERTRAK supports IBM & Compatibles using DOS 2.0 or later, sells for $295.00 and has a 30 day money back guarantee. Free info & demo.

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800) 227-0644 (816) 776-2700
FAX (816) 776-8398

Number 310 on the Reader Service Card

Programmer Info Cards

MICRO CHART plastic reference cards are full page, professional, 8.5 x 11", time-saving summaries packed with programming info. 100% plastic, colorful, 2 sides. Clears away books, speeds work. 14 day money-back guarantee.

MC-CL	"C" language with library functions
MC-IBAS	Introduction to BASIC programming
MC-IBM	Complete IBM Basic
MC-SHEL	UNIX Shell programming
MC-ALG	Useful BASIC subroutines
MC-PS	Postscript® reference
MC-68K	68000 assembly lang.
MC-8086	8088 & 8086 assembly
MC-Z80	Z80 assembly lang.
MC-7400	7400 TTL pinouts

only $5.95 each

To order: call (800) 342-5930 with Visa/MC or send check to Micro Logic, POB 70, Dept M100, Hackensack, NJ 07602. $5.95 each. Add $2 S&H any quantity.

Number 337 on the Reader Service Card

FREE CATALOG!

LATEST AWARD BIOS
PC/XT, 286, 386

COMPILED BASIC TOOLS
Libraries, Utilities
for QuickBASIC 4.5

HARDWARE UPGRADES
for PCs & PC/XTs

(800) 423-3400 - (412) 782-0384
KOMPUTERWERK INC.
851 Parkview Blvd.
Pittsburgh, PA 15215

Number 155 on the Reader Service Card

RELATIONAL DATABASE LIBRARY

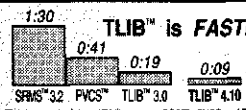
For TurboC and Microsoft C on DOS systems. The 114 functions in the **"c_wndw"** and **"c_ndx"** libraries support:

- **Multiple B-tree index files**
- **Stacked, overlapping windows**
- **dBase .DBF & .NDX file formats**
- **Multi-level file relationships**
- **Pull-down and pop-up menus**
- **Hashed random file access**
- **Data dictionary field access**

Libraries $99.50; Source $199.50; VISA/MC

Marietta Systems, Inc.
P.O. Box 71506
Marietta, GA 30007
(404) 565-1560

Number 223 on the Reader Service Card



TLIB™ is *FASTEST!*

Times are to update a 45K library on a PC/XT. PVCS and TLIB 3.0 are from Sept 87 *PC Tech Journal*. SRMS and TLIB 4.10 are later versions.

TLIB™ is *BEST!*

"Do not be fooled by the fact that this is the least expensive of the five packages reviewed here - TLIB has features and power to spare" John Rex, **Computer Language**

"TLIB is a great system" J. Vallino, **PC Tech J**

- **Full-Featured Version Control for Software Professionals.** Check-in/out locking. Branching. Keywords. Wildcard and list-of-file support. Can merge parallel changes and undo intermediate revisions. Network and WORM support. **Mainframe compatible** deltas for Pansophic, ADR, IBM, Sperry. Integrated Landon Dyer PD **MAKE**.

MS-DOS 2.x, 3.x, 4.x ***Just $99.95*** + $5 s/h Visa/MC
5 station LAN license $299.95 + $5 s/h, call for other sizes

BURTON SYSTEMS SOFTWARE
PO Box 4156, Cary, NC 27519 **(919) 856-0475**

Number 288 on the Reader Service Card

PC-lint

C Programmers: Locate bugs, glitches, quirks and inconsistancies in your C code. PC-lint supports multiple modules, K&R, ANSI. Specify OS/2 or MS-DOS. Only $139.

Gimpel Software
(215)584-4261
3207 Hogarth Ln
Collegeville PA 19426
30 Day Money-back Guarantee

Contact this advertiser directly

Advanced **Programmer's Books & Tools for MS-DOS & OS/2**

Professional books, function libraries, and utilities for creating high-performance MS-DOS programs, TSRs, OS/2 programs, and Presentation Manager applications.

C and Assembly Language!

Please call or write for free catalog:

Michael J. Young
Young Software Engineering
20 Sunnyside Avenue, Suite A, Dept. PJ02
Mill Valley, CA 94941
415/383-5354

Number 166 on the Reader Service Card

PASCAL OBJECT LIBRARIES
For Turbo 5.5 & Quick Pascal 1.0

Andrews Data Systems introduces OOP tools for Pascal. Prototype sophisticated user interfaces quickly and easily. Create complex simulations and real-time systems in a fraction of the time required for procedural languages.

Windows - Form/field support and real-time extensions (lifeline and animated bar graphs). Text and graphic mode support. Includes form builder and attribute select.

Remote Object Support - distributed processing over serial line or network. Includes serial library.

Concurrent Object Support - multi-task within your application.

$59 each or $89 with source. Serial library alone: $39 or $59 w/source. Windows/Remote/Concurrent: $149 or $199 with source. Add $5 S/H all orders. Check, MO, VISA/MC. No CODs.

ANDREWS DATA SYSTEMS
P.O. BOX 37123
DENVER, CO 80237
1-800-255-2550 ext. 615

Number 318 on the Reader Service Card

epsilon programmer's editor has what you've been missing . . .

State of the art editor for DOS, OS/2, Xenix & Unix for PC/XT/AT's

Undo and Redo functions. Ultimate customizability with C-like extension language with source code to all commands. Compiler journaling. EMS support. Multi-tasking compiles. Unlimited file size, number of files. Full command line editing. Record keystroke sequences. Optional backups. Not copy protected.

Ask about our free trial!

$195 direct 60 day money-back guarantee

Lugaru Software Ltd.
5843 Forbes Avenue
Pittsburgh, PA 15217
(412) 421-5911
FAX: (412) 421-6371

Number 271 on the Reader Service Card

Network API Library

For *C* or *Pascal*, supports both *Microsoft* and *Turbo* compilers, Compatible with *IBM, 3Com, Banyan*, and *Novell* Networks. Includes DOS network functions, **NetBIOS, IPX/SPX,** Client/Servers, **Remote Procedure Calls,** Remote screens/windows. $99.95

NetWorkStation Utilities

Network Consulting, Distributed Processing, Classroom Screen Broadcasts, Work Groups, Device Sharing. NetBIOS/Netware $99.95

Information Software Associates
Box 2582, Iowa City, IA 52244
Call for more info (319) 354-6804

Number 294 on the Reader Service Card

COLLEGE DEGREES

. . . from fully-accredited Colleges and Universities. We help Computer Professionals avoid years of unnecessary class work. Get the job offers and recognition you deserve. Phone UDA for more information and our free booklet "Career Tactics and Strategies."

University Degree Advisory
☎ 800-765-7272

Number 311 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programmer's Journal

March/April 1990
Vol. 8.2

Advertising Space Reservation

Deadline:
January 5, 1990

Materials Due:
January 12, 1990

Please call *Programmer's Journal* to reserve your space.

503-747-0800

1000'S OF NEW COMPUTER BOOKS UP TO 40% OFF

New titles for the professionals including books for PCs and compatibles, MAC, UNIX, minis, and mainframes. Dos, OS/2, X Window, X Open, and UNIX V/386, Ada, C, C++, Modula-2, Pascal, LISP, FORTH, ASM, and other languages.

Desktop publishing, word processing, CAD, graphics, image processing, and hardware.

Send five 25¢ stamps for complete 70-page catalogue.

Computer Books
P.O. Box 1758
Solano Beach, CA 92075 **RS# 340**

dBASE FILE ACCESS FROM C

Code Base 4 is a library of C routines which gives complete dBASE and Clipper functionality and file compatibility. The routines correspond directly to dBASE commands. Create multi-user applications using pulldown menus, popup windows and data entry.

Code Base 4 supports DOS, SCO Xenix, OS/2 and Microsoft Windows.

$295 with Source! FREE DEMO.

Sequiter Software Inc.
P.O. Box 5659, Station L
Edmonton, Alberta
Canada T6C 4G1
403-439-8171
Fax: 403-433-7460 **RS# 269**

CSOURCE APPLICATION GENERATOR

C code WYSIWYG screen, report, & program generator. Library code may be enhanced for custom generation. Supports multi-user development & DOS 3.1/Novell networks. Produces small-sized DATA ENTRY, REPORT, MENU, INDEX REBUILD & FILE RESTRUCTURE programs with multiple screens & files. Very fast development & execution speeds. Requires 256K RAM, DOS 2.1 or later, Faircom Ctree or Novell Btrieve. For Turbo or Microsoft C 5.0. No royalties, ALL generated applications source is supplied.

$395.00 executable, $595.00 full source code. Multi-programmer licenses.

MBAC
13320 Castana Ave.
Downey, CA 90242
213-531-7259 **RS# 291**

PJ CODE IS AVAILABLE

ON DISK
$10.00 each, VISA/MC
CALL: 1-800-234-0386

ON BBS
See ad on page 16
for BBS downloads.

DO IT YOURSELF!
XenoCopy-PC
PC-DOS program lets your PC *Read/Write/Format* *over 350 formats* **$79.95**

XenoFont
high quality text screen printouts
$49.95

+ $5.00 S/H + Tax if CA.
XenoSoft • (415) 644-9366
2210 Sixth Street,
Berkeley, CA 94710

Number 123 on the Reader Service Card

"QUICK & CLEAN" COBOL

The COBOL Report Activator (COBRA):

- Generates efficient, structured COBOL programs
- Is easy to learn and use
- Accepts your COBOL code and COPY members
- Produces audit trail reports
- Up to 5 output files per program
- Provides report options & site defined defaults
- Is available for Mainframe & PC

COBRA
Systems Software
David Olivadoti, P.O. Box 750817
Petaluma, CA 94975
(707) 763-5123

Number 281 on the Reader Service Card

QuickBASIC File Indexing

Create B+ Tree file indexes in QuickBASIC with **Index Manager.™** Access files randomly by full or partial key, or sorted forward or backward. One module performs all functions making IM very easy to use. Only indexes managed! You retain full control over your data. Assembler code and cache buffers make IM very fast. Demo on GEnie and CompuServ. Eight sample programs and 64 page Users Guide included. Only $59 (+tax in CA). VISA/MC accepted.

CDP Consultants
1700 Circo Del Cielo Drive
El Cajon, CA 92020
(619) 440-6482

Number 335 on the Reader Service Card

Double-decker STAND
$124.99
Sturdy, tubular steel, enamel finished oak and putty.

Piggybacks two printers. Casters optional- $8.00.
24x22x41¾High

Single printer stands- tubular steel, oak top. * Includes casters. Discounts avail.
132 col. 24x14x26½H. . . . $73.00*
80 col. 16x14x26½H. . . . 59.99
CRT workstations/tables avail., call for prices. 214 / **299-6658**

TRI-STAR DISTRIBUTION
3816 Flamingo Ln., Irving, TX 75062

Number 314 on the Reader Service Card

Why you want BATCOM!

BATCOM is a batch file compiler that compiles your ".bat" files to ".exe" files to make them faster, more professional, and more capable. **BATCOM** extends DOS with new commands so you can read keyboard input, perform arithmetic, use subroutines, and much more. In addition, **BATCOM** protects your source code, and you can distribute your compiled programs without royalties. Only **$59.95**

ORDER TODAY!

Wenham Software Co.
5 Burley St.
Wenham, Ma. 01984
(508) 774-7036

Number 320 on the Reader Service Card

PCHRT V3.00
HIGH RESOLUTION TIMER TOOLBOX

The definitive answer to execution profiling and embedded timer problems. 30 functions manage 100 software timers with microsecond accuracy. Self calibrating. Extensive and flexible timer reporting. Profiles selected ROM BIOS and MSDOS interrupts.

Supports TP 4 & 5, TC, MSC 5.x and QC 2.0. Libraries, manual, examples, and full source included.

$49.95 ppd USA, elsewhere add $4.00, Mich add 4%. VISA/MC

RYLE DESIGN
P.O. Box 22
Mt. Pleasant, MI 48804
517-773-0587

Number 210 on the Reader Service Card

dCURSES

A full implementation of the UNIX Curses Library for MS-DOS.

MSC
Turbo C
Lattice C

Object $69.95
Source $149.95
VISA/Mastercard
Call for FREE Shareware Copy

HIGH ASPECT DEVELOPMENT
9821 South Pulaski
Evergreen Park, IL 60642
(312) 424-5997

Number 313 on the Reader Service Card

MIDNIGHT ENGINEERING™

A new magazine for entrepreneurial hardware and software engineers that will encourage and challenge you to personally develop and market your own products.

- **PRACTICAL ARTICLES**
- **INSIGHTFUL INTERVIEWS**
- **DETAILED PRODUCT REVIEWS**

This is your chance to influence a brand-new magazine. *PJ* readers are invited to get a **FREE** copy of the premiere issue of *Midnight Engineering* by calling or writing us with ideas.

DATArx
111 E. Drake Rd., Suite 7041
Fort Collins, CO 80525
303-491-9092

Number 272 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PERFECT 360K FLOPPIES
using your 1.2M drive

How many times have you carefully formatted a 360K floppy in a 1.2M drive, then written data to it, only to find it unreadable in a real 360K drive? Too many times, no doubt!

Send $10 (+ $1 shipping) for the booklet **Perfect 360K Floppies Every Time,** describing a simple 100%-effective solution requiring **no extra software or hardware.** Put to use immediately, saving time and money. VISA/MC.

Objective Systems & Technologies
133 E. De La Guerra, Suite 423-P
Santa Barbara, CA 93101
805-564-8125

Number **338** on the Reader Service Card

UNDERSTANDING C

6 hours of videotape gives the in-depth details of the ANSI C language. 200 screens of examples teach you to write like an experienced pro. Learn about structs, pointers, bit fiddling—all the details of C syntax. 100% satisfaction guaranteed! $59.95 (no extra charge for C.O.D.) Call now for info or C.O.D. order.

1-800-762-5515
Automation Education
2309 Royce Dr.
Arlington, Texas 76016-1232

Number **207** on the Reader Service Card

OPT-TECH SORT/MERGE

Extremely fast Sort / Merge / Select utility. Run as an MS-DOS command or CALL as a subroutine.
Supports most languages and filetypes including Btrieve and dBase. Unlimited filesizes, multiple keys and much more!

MS-DOS **$149.**
OS/2, UNIX **$249.**

Opt-Tech Data Processing
P. O. Box 678
Zephyr Cove, NV 89448
(702) 588-3737

Number **16** on the Reader Service Card

BTRIEVE ALTERNATIVE

Application programmers: Get a secure, multilanguage, multiuser, full, flexible indexed file manager. AXOS, only $395. "AXOS provides 57 functions, including audit trail and key-only file management functions, and 64 index paths vs. Btrieve's 31 operations and 24 access paths."
—*PC TECH JOURNAL* (Oct 1988)

McSoft, Inc.
P.O. Box 12044
Atlanta, GA 30355
404-984-0608 **RS# 230**

WHITNEY EDITOR $49

Small and fast. Uses all available memory. Split-screen editing. Configurable keyboard. Regular expression search. One-key compile. Features for writing documentation. Condensed/Outline display. Runs on IBM PC's, AT's, and PS/2's.

USA shipping and handling $3. Outside USA $15. CA residents add sales tax.

Whitney Software, Inc.
P.O. Box 4999
Walnut Creek, CA 94596
415-933-9019 **RS# 298**

FREE TRIAL

Pinnacle File Manager

We guarantee immediate productivity gains from using our C file manager library. Simple table-and-column view of data with advanced B-Tree indexing and hi-speed cache frees your brain to solve application-specific problems.

Vermont Database Corp.
RR3 Box 6516
Stowe, VT 05672
802-253-4437

Number **328** on the Reader Service Card

sPORTt serial PORT tester

sPORTt is a serial port tester for people who don't want to be data communications experts, but use or write software for serial devices, such as plotters, mice, digitizers, modems, serial printers, or computer to computer links.

- **displays all data going in and out**
- **displays ASCII, decimal, hex, binary**
- **lets you choose handshaking lines**
- **sends/receives disk files**
- **automatically runs port diagnostics**

Order today only $89.95!

WHS
3037 Grass Valley Hwy #8201
Auburn, CA 95603
916-885-2480

Number **329** on the Reader Service Card

Create Windows Applications Fast

WinTrieve ™

- Powerful data management
- Indexed file support (ISAM)
- C & Actor libraries
- Only $395; no royalties

800-869-1144
or 708-328-3800

The Whitewater Group ®

Number **73** on the Reader Service Card

ADVERTISER INDEX

Reader Service	Advertiser Name	Page Number
305	Acme Software	71
282	Acucobol Inc.	12
331	Addison-Wesley Pub. Co.	41
318	Andrews Data Systems	94
332	At Last! Software Inc.	12
30	The Austin Code Works	55
207	Automation Education	96
51	Az-Tech Software, Inc.	94
310	Az-Tech Software #2	94
303	BBI Computer Systems	11
244	Block Island Technologies	71
3	Borland International	C4
288	Burton Systems Software	94
321	Catenary Systems	25
299	Clarion Software	1
*	CNS, Inc.	51
281	COBRA Systems Software	95
340	Computer Books	95
277	Covox Inc.	30
335	CDP Consultants	95
272	DATArx	95
308	Daytron Electronics, Inc.	17
229	Disk Software, Inc.	27
316	EDI	C3
307	F.G. Soft Company	25
333	FTP Software	35
261	Genus Microprogramming	27
*	Gimpel Software	94
323	Glenco Engineering Inc.	44
334	Hamilton Laboratories	34
248	Hammerly Computer Services, Inc.	92
293	Hammerly Computer Services, Inc.	93
42	Harvard Softworks	54
258	Helios	43
313	High Aspect Development Corp.	95
294	Information Software Associates	94
226	Ithaca Street Software	30
295	Knowledge Dynamics Corporation	17
155	Komputerwerk Inc.	94
63	Laboratory Microsystems Incorporated	30
44	Lahey Computer Systems Inc.	36
246	Logical Alternatives Incorporated	11
271	Lugaru Software Ltd.	94
223	Marietta Systems Inc.	94
291	MBAC	95
230	McSoft	96
337	Micro Logic Corporation	94
232	Microsoft Human Resources	9
104	Microway	53
121	MISOSYS, Inc.	31
266	Mix Software	18
94	MMC AD Systems	69
319	Mt. St. Helens Software	7
312	MTD Software Development Co. Ltd.	10
109	Nanosoft	34
278	Netwise	38
302	NTERGAID Inc.	59
160	Nu-Mega Technologies	3
338	Objective Systems & Technologies	96
16	Opt-Tech Data Processing	96
*	Paradigm Systems	29
208	PC Tech	25
6	The Periscope Company, Inc.	37
*	***PJ Code DiskScription***	8
279	Pocket Soft, Inc.	19
*	***Programmer's Journal*** Mail Order Store	88
97	Qualstar Corp.	29
147	Quarterdeck Office Systems	C2
296	Que Corporation	22
324	Rogue Wave Software Associates	11
210	Ryle Design	95
26	Sage/Polytron Corporation	14
269	Sequiter Software Inc.	95
165	SLR Systems	18
98	The Software Engineering Store	78
198	Software Science Inc.	87
221	Software Security, Inc.	5
110	Softway, Inc.	26
10	South Mountain Software	13
330	Stanwood Associates Inc.	24
297	Talking Technology, Inc.	6
289	Technolock Software, Inc.	91
166	The Tool Makers	59
327	Tools & Techniques Inc.	17
314	Tri-Star Distribution	95
276	Tri-Technology Systems Inc.	54
25	TurboPower Software	51
309	UnderWare, Inc.	42
311	University Degree Advisory	94
125	V Communications, Inc.	59
341	V Communications, Inc.	77
328	Vermont Database Corporation	96
287	Vestronix	67
304	Visitech Software	25
320	Wenham Software Co.	95
100	Western Wares	51
339	W.H. Roetzheim & Associates	17
73	The Whitewater Group	96
298	Whitney Software, Inc.	96
329	WHS Western Hydro Systems	96
123	XenoSoft	95
168	Young Software Engineering	94

When contacting any of the above advertisers, please tell them you saw their ad in ***Programmer's Journal*** and that you appreciate their continuing support of this magazine. The advertisers with an * prefer to be contacted directly. ***PJ***'s Advertiser Index is provided as a free service, and ***Programmer's Journal*** cannot be held responsible for omissions or errors.

PRODUCED 2004 BY CIMTZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

EARN ROYALTIES NOW

ON THE SHAREWARE YOU'VE DEVELOPED

On October 1st a new on-line service was born. Called EDI, it is an on-line service where shareware enthusiasts go to download shareware. And it is a place where shareware authors, like you, can go to distribute the shareware you've developed.

Unlike other shareware distributors, EDI provides royalties to its authors. Royalties are provided every time a copy of your software is downloaded and every time anyone enters your optional on-line customer support section. Every time.

To attract huge numbers of shareware enthusiasts to EDI (and earn you more royalties), we provide shareware enthusiasts with access to EDI over a growing number of on-line networks.

So, no matter what type of share ware product you offer, we'll help you reach enthusiasts all around the country and all around the world.

As a software developer we invite you to take advantage of our services and use it to distribute your shareware products. There is no fee for submitting a product (other than the network connect-time fee). We will gladly review your product. And if it is accepted, we will make your product available to our callers.

So give us a call. And find out how you can start earning royalties now.

Instructions (MSC Network)

Set your modem to 7-E-1 and dial one of the numbers below. Once connected press [return] [return] to get a # prompt and press X[return] to get a (star) prompt and type .vtt [return] to enter the MCS network. Follow the registration instructions and once on-line press EDI for instant access.

EDI National Access (MSC Network)

ALABAMA	
Anniston	(205)820-7609
Birmingham	(205)254-3120
Montgomery	(205)279-0314
ARIZONA	
Phoenix	(602)233-9592
ARKANSAS	
Little Rock	(501)376-1535
CALIFORNIA	
Burbank	(818)954-9161
Corona	(714)736-6820
El Monte	*(818)442-9291*
Garden Grove	(714)895-4946
Los Angeles	(213)772-6971
Monterey	(408)899-7855
Mountain View	(415)964-8240
Newport Beach	(714)895-4946
No. Hollywood	(818)786-4111
Sacramento	(916)452-7356
San Bernardino	(714)888-7611
San Diego	(619)293-3620
San Fransisco	(415)391-9080
San Jose	(408)554-0344
Van Nuys	(818)786-4111
COLORADO	
Colorado Springs	(719)540-8220
Denver	*(303)893-3803*
CONNECTICUT	
Hartford	(203)728-6766
Milford	(203)728-6766
New Haven	(203)877-2751
Stamford	(203)327-6745
Wilmington	(302)571-1433
D.C.	
Washington	(301)937-0530
Washington	(301)937-0722
FLORIDA	
Fort Lauderdale	(305)462-3104
Jacksonville	(904)398-2074
Miami	(305)651-2387
Pensacola	(904)438-7203
Tampa	(813)626-5235
GEORGIA	
Atlanta	(404)256-9151
Augusta	(404)790-5406
Columbus	(404)682-1103
HAWAII	
Honolulu	(808)524-7950
ILLINOIS	
Chicago	(312)704-0313
Oak Brook	(312)572-8550
Oak Brook	(312)572-8566
Rock Island	(319)324-0042
Springfield	(217)525-6683
INDIANA	
Indianapolis	*(317)634-4301*
IOWA	
Davenport	(319)324-0042
Des Moines	(515)224-0011
KENTUCKY	
Louisville	(502)587-9560
LOUISIANA	
New Orleans	(504)566-0002
MARYLAND	
Baltimore	(301)760-1950
MASSACHUSETTS	
Boston	(617)965-4900
MICHIGAN	
Detroit	(313)963-9141
Grand Rapids	(616)456-6047
MINNESOTA	
Minneapolis	(612)338-5540
MISSOURI	
Kansas City	(816)283-3719
St. Louis	(314)721-4801
NEBRASKA	
Omaha	(402)341-2425
NEW JERSEY	
Boonton	(201)335-2332
Cherryhill	(609)235-3220
Hackensack	(201)489-9020
Moorestown	(609)235-3220
New Brunswick	(201)846-9310
Newark	(201)564-6300
Parsippany	*(201)335-2332*
Springfield	(201)564-6300
Trenton	(609)883-5113
Whippany	(201)335-2332
NEW YORK	
Buffalo	(716)896-5410
Long Island	(516)481-8028
New York	(212)391-1920
New York	(212)391-0370
Poughkeepsie	(914)454-1808
Purchase	(914)694-6842
Syracuse	(315)428-1674
NORTH CAROLINA	
Charlotte	(704)374-0587
Fayetteville	*(919)436-0145*
Raleigh	(919)821-7535
NORTH DAKOTA	
Fargo	(701)237-5282
OHIO	
Akron	(216)535-6635
Cincinnati	(513)241-1446
Cleveland	(216)621-3880
Columbus	(614)461-8450
OKLAHOMA	
Lawton	(405)355-0275
Oklahoma City	(405)272-9751
OREGON	
Portland	(503)241-0549
PENNSYLVANIA	
Harrisburg	*(717)236-5241*
Philadelphia	(215)829-0912
Pittsburgh	(412)261-9305
SOUTH CAROLINA	
Columbia	(803)771-7665
SOUTH DAKOTA	
Sioux Falls	(605)334-1832
TENNESEE	
Memphis	(901)521-0529
Nashville	(615)292-6801
TEXAS	
Austin	(512)499-8029
Dallas	(214)638-7850
Fort Worth	(817)460-3711
Houston	(713)981-9797
Kileen	(817)532-8980
San Antonio	(512)525-1692
UTAH	
Salt Lake	(801)544-0086
VIRGINIA	
Newport News	(804)874-5121
Norfolk	(804)858-2715
Richmond	(804)788-0401
WASHINGTON	
Seattle	(206)624-3580
Spokane	(509)455-9398
WEST VIRGINIA	
Charleston	(304)345-7505
Parkersburg	(304)485-7569
WISCONSIN	
Madison	(608)255-5857

EDI International Access (MSC Network)

AUSTRALIA	
Brisbane	07-252-4132
Brisbane	07-252-4124
Melbourne	*03-696-5523*
Sydney	02-647-9847
BELGIUM	
Brussels	02-648-0710
Brussels	02-647-9847
CANADA	
Calgary	(403)234-7841
Montreal	(514)282-0222
Ottawa	(613)563-9085
Toronto	(416)438-7570
Vancouver	(604)669-4040
DENMARK	
Copenhagen	1-822015
FINLAND	
Helsinki	92919
FRANCE	
Paris	1-43-352327
Paris	1-43-207441
Paris	1-43-350101
GERMANY	
Frankfurt	069-6666062
Frankfurt	069-6663062
Frankfurt	069-6666881
Frankfurt	069-6668297
Frankfurt	069-6666886
HONG KONG	
Hong Kong	5-202525
Hong Kong	5-202857
ITALY	
Milan	2-545-6351
Milan	*2-545-8840*
Milan	2-545-9145
Milan	2-545-8649
Milan	2-540-0425
Milan	2-545-0620
Milan	2-546-8994
JAPAN	
Osaka	6-202-4159
Osaka	6-202-7970
Tokyo	3-234-9251
Tokyo	3-234-8275
MEXICO	
Mexico City	905-538-7645
Mexico City	905-538-4252
NETHERLANDS	
Amstelveen	*020-417855*
Amstelveen	020-476171
NORWAY	
Oslo	02-121496
Oslo	02-121795
SOUTH AFRICA	
Durban	031-290277
Jahannesburg	011-403-5730
SPAIN	
Madrid	34-1-742-8244
Madrid	34-1-742-8210
SWEDEN	
Stockholm	8-834095
Stockholm	8-834090
Stockholm	8-764-6595
SWITZERLAND	
Bern	031-260931
Bern	031-260787
Bern	031-260691
Geneva	22-798-5756
Geneva	22-798-6364
UNITED KINGDOM	
London	01-427-4393
London	01-439-4055

ELECTRIC DISTRIBUTION INTERNATIONAL

EDI is a service of ***Electric Distribution International***
1332 Hermosa Avenue, Hermosa Beach, CA 90254.
For customer support call (800) 776-6076
Outside the continental United States call (213) 374-3478

Number **316** on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

BORLAND'S TURBO C PROFESSIONAL 2.0

CORPORATE TOP RATED PC WEEK 1989 SATISFACTION POLL

PC WEEK POLL: C COMPILERS

	Overall Weighted Score	Overall Reliability	Complete of Command Descript.	Overall Perform.	Complete & Organiz. Document.	Document Clarity	Compiling Process Efficiency	Product Support Quality	Value Relative To Cost	Product Support Access.
Turbo C 2.0 (Borland International)	81	87	79	84	77	78	86	72	70	93
C Optimizing Compiler 5.1 (Microsoft Corp.)	76	83	80	81	78	74	76	68	67	70
C++ 1.07 (Zortech Inc.)	66	68	64	71	63	63	69	60	58	76

"Microsoft was No. 1, but they have been unseated by Borland." PC Week, *May 8, 1989*

PC WEEK POLL: SOFTWARE DEBUGGERS

	Overall Weighted Score	Overall Reliability	Effective. Programmer Interface	Document. Clarity	Complete. Command Descript.	Complete. & Organize. Document	Overall Perform.	Integration Within Programming Environment	C Compiler Compatibility	Product Support Quality	Product Support Access	Value Relative To Cost
Turbo Debugger 1.0 (Borland International)	84	89	90	81	81	81	89	88	81	73	72	93
Codeview 2.2 (Microsoft Corp.)	73	80	71	72	74	74	74	74	78	67	64	72

"Borland's Debugger outshines Microsoft's Codeview." PC Week, *May 15, 1989*

It's two winners in one.

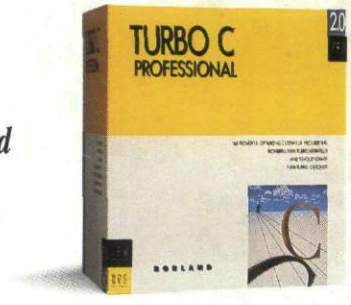
Turbo C,® the core of Turbo C Professional, was the outright winner in *PC Week*'s Poll of Corporate Satisfaction on C compilers. Overall, Borland won with 81. Microsoft® placed second.

Turbo Debugger,® also included in Turbo C Professional, was the outright winner in EVERY category in *PC Week*'s Poll Of Corporate Satisfaction on Debuggers. And, once again, we topped the score with 84, overall. Microsoft came in second-best, 11 points behind.

Get Borland's Turbo C Professional and get the best of both worlds: our top-rated C compiler and our top-rated Debugger.

Call **(800) 345-2888*** and we'll send you both *PC Week* polls and technical specifications on Turbo C and Turbo Debugger.

Turbo C Professional includes both Turbo C 2.0 and Turbo Assembler® & Debugger.



BORLAND

CODE: MC44

*From Canada, Call (408) 848-4391. Reprinted from PC Week, May 8, 1989; PC Week, May 15, 1989. Copyright ©1989 Ziff Communications Company. Turbo C, Turbo Debugger, and Turbo Assembler are registered trademarks of Borland International. Copyright ©1989 Borland International, Inc. All rights reserved. BI 1328

Number 3 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED